PATENT ABSTRACTS OF JAPAN

(11)Publication number : 06-266334
(43)Date of publication of application : 22.09.1994

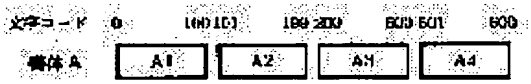
(51)Int.Cl. G09G 5/22
G06F 15/72

(21)Application number : 05-049043
(22)Date of filing : 10.03.1993

(71)Applicant : CANON INC
(72)Inventor : SAKAI TETSUO
SASAKI YASUHIKO
YAMAGUCHI HIROSHIGE
KASHIWAGI MASAKI
ISHIGURO KENJI
OSADA MAMORU
MATSUKI HIROSHI

(54) FONT DATA CONTROL METHOD AND CHARACTER PATTERN GENERATING METHOD

(57)Abstract:
PURPOSE: To provide the control method for font data which is flexible.
CONSTITUTION: When the whole character space, which constitutes one font, is, for example, from 0 to 500, the space is divided into several areas. For example, the space is divided into code ranges 0-100, 101-199, 200-500, and 501-600, and font data file by the space areas are controlled. Therefore, when codes 200-500 are not used, only the part can be excluded from the control and the file of the part can also be deleted.



LEGAL STATUS

[Date of request for examination]	25.12.1997
[Date of sending the examiner's decision of rejection]	04.07.2000
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]	
[Date of final disposal for application]	
[Patent number]	3285995
[Date of registration]	08.03.2002

[Number of appeal against examiner's decision of rejection] 2000-12142

[Date of requesting appeal against examiner's decision of rejection] 03.08.2000

[Date of extinction of right]

CLAIMS

[Claim(s)]
[Claim 1] The typeface data control approach which divides character code space into two or more partial code space, and is characterized by managing the typeface data file which became independent for every divided character code range in the typeface data control approach for taking out the data about the demanded alphabetic character from a typeface data file.
[Claim 2] The typeface data-control approach which divides character code space into two or more partial code space, manages the typeface data file which became independent for every divided character code range, detects the frequency of the alphabetic-data income demand for every character code space, and is characterized by to change the retrieval sequence of each typeface data file in order of the frequency of each detected character code space in the typeface data-control approach for taking out the data about the demanded alphabetic character from a typeface data file.
[Claim 3] The typeface data control approach which divides character code space into two or more partial code space, and is characterized by making it subordinate to the typeface data file of the typeface of others at least one of the partial typeface data file which manages the typeface data file which became independent for every divided character code range, and constitutes attention typeface data in the typeface data control approach for taking out the data about the demanded alphabetic character from a typeface data file.
[Claim 4] The typeface data control approach which divides character code space into two or more partial code space, and is characterized by carrying out storage maintenance and managing the storing logical memory device of each typeface data file which became independent for every divided character code range in the typeface data control approach for taking out the data about the demanded alphabetic character from a typeface data file.
[Claim 5] The typeface data control approach which divides character code space into two or more partial code space, and is characterized by managing the typeface data file which became independent for every divided character code range, and managing the typeface data file of different character sets in the same character code space as said 1st typeface similarly to the 1st typeface in the typeface data control approach for taking out the data about the demanded alphabetic character from a typeface data file.
[Claim 6] In the typeface data control approach which takes out the data about the alphabetic character demanded from the typeface data file To at least one of each typeface data files of the which constitute the 1st typeface which manages the typeface data file which divided character code space into two or more partial code space, and became independent for every divided character code range The typeface data control approach characterized by making the typeface data file of the character code space where it corresponds in the 2nd typeface correspond, and managing.
[Claim 7] Since one typeface is specified in the typeface data control approach for distinguishing the classification of typeface data, As information for specifying allocation and one typeface as the information which expresses the size of the line in a style, a character width, and a typeface at least at the bit of a predetermined number A style, a character width, the typeface data control approach characterized by building by one data having each size of the line in a typeface.

[Claim 8] The typeface data control approach characterized by encoding the typeface name for specifying one typeface, a style, width of face, size, and the character code range to support to a notation alphabetic character, and making the name length of the limited file correspond out of two or more typefaces in the typeface data control approach of managing the file about typeface data.
[Claim 9] In the character-pattern generating approach of generating the demanded reduced-character pattern from the character pattern to which predetermined dot size corresponds The number of the line which performs the deletion or superposition determined with said demanded size to the character pattern of the dot size of said typeface or/, and trains more than a predetermined number In a certain case The character-pattern generating approach characterized by repeating said contraction processing until it carries out contraction processing as deletion or superposition of said predetermined number and the target reduced-character pattern is generated by this contraction processing.
[Claim 10] In the typeface data control approach of managing the data about an alphabetic character as a typeface data file When registering with the typeface file which specified one typeface data file by the typeface name and the typeface number, and the user created at a system When the typeface file of the same typeface name is already registered The typeface data control approach characterized by assigning an intact number when [concerned] it registers, and the good typeface name which **** is changed and the typeface file of the same typeface number is registered into the system, and a system updating the contents of the predetermined file which manages a typeface.

[Translation done.]

DETAILED DESCRIPTION

[Detailed Description of the Invention]
[0001]
[Industrial Application] This invention relates to the generating approach of the character pattern generated as the typeface data control approach and it which manage the data file of the various typefaces used with the typeface data control approach and the character-pattern generating approach, for example, a personal computer, a word processor, desktop-publishing equipment, etc. are also.
[0002]
[Description of the Prior Art] Conventionally, the typeface data managed in a character-manipulation system stored the data of a whole sentence character per typeface in one file from the reasons of the ease of management etc.
[0003]
[Problem(s) to be Solved by the Invention] There is a problem shown below by the typeface data control approach by the above-mentioned conventional approach.
[0004] All the typeface data are beforehand stored in the file, and the object of install had only selection in a typeface unit to the last. For this reason, even the typeface data which do not not much have being used generally will be installed in coincidence, and it had resulted in pressing disk memory.
[0005]
[Means for Solving the Problem] This invention tends to be accomplished in view of this conventional technique, and it is going to offer the management method of supple typeface data.
[0006] For this reason, the one typeface data control approach of this invention is equipped with the stroke shown below, for example. That is, in the typeface data control approach for taking out the data about the demanded alphabetic character from a typeface data file, the typeface data file which divided character code space into two or more partial code space, and became independent for every divided character code range is managed.
[0007]
[Function] In the configuration in the configuration of this this invention, one typeface data is divided into plurality in character code space, and, moreover, has the file of typeface data for every divided partial code space of the.
[0008]
[Example] Hereafter, the invention in this application is explained to a detail with reference to a drawing.
[0009] First, in case the system treating alphabetic data, such as a personal computer, performs a display or printing of an alphabetic character, a motion until it gains typeface data is explained briefly.
[0010] Drawing 1 is the block diagram showing a fundamental configuration for the system of these above to operate.
[0011] In drawing 1 , 101 is CPU, i.e., a central processing unit, and performs starting of the above-mentioned whole system, data processing, etc. 102 is a read-only memory (ROM) and is storage regions, such as a system bootstrap (boot program) and character-pattern data. 103 is random

access memory (RAM), and it is used for storage temporarily [various data] while a system program, various application programs, the driver program for character-pattern generating, etc. are loaded. Of course, the program concerning the flow chart mentioned later is also loaded to this RAM103, and is performed. 104 is a keyboard control section (KBC) and transmits key input data to reception and CPU101 from the keyboard (KB) of 105. 106 is a CRT control section (CRTC) which controls a display unit (CRT) 107. 109 is external storage, such as a floppy disk drive unit (FD) and a hard disk drive unit (HD), and various programs and font data including a system program or an application program are stored in these. The program memorized by these external storage 109 will be loaded to RAM103, and will be performed by CPU101 at the time of program execution. 110 is a printer control section (PRTC) and 111 is a printer (PRT) which forms the visible image based on the data sent out under control of PRTC110 in the record paper. 112 is a system bus and should become the path of the data between above-mentioned components.
[0012] Drawing 2 is drawing having shown the memory map of RAM103. 201 is a system area where a system is loaded, 202 is a field where the alphabetic character expansion processing section is loaded, and 203 is a field where the alphabetic character expansion processing section stores bit pattern data. 204 is a field read in order to refer to the vector pattern data stored on the external storage of 109 at the time of alphabetic character expansion processing. 205 is a field for generating a management information table for said system operating. 206 is the cache field of typeface vector data.
[0013] Drawing 3 is drawing showing the structure of a system treating said alphabetic data.
[0014] In drawing 3 , 301,302,303 expresses various typeface data memorized in the external storage region 109 in drawing 1 . A limit does not have substantially the class of typeface data memorized by external storage 109, and its number.
[0015] 304 shows the system control section and 201,204,205 in drawing 2 is contained. The alphabetic character expansion processing section, i.e., a rasterizer, is shown, only software may realize the alphabetic character expansion processing facility, hardware may be realized, and even if 305,306 uses the both sides of hardware and software, it is not cared about. Moreover, if at least one number of the rasterizers contained in said system is contained, there will be no other limits. 307 expresses the application section (for example, program of a word processor etc.) which publishes an alphabetic character acquisition demand to a system.
[0016] In drawing 3 , the application section 307 emits an alphabetic character acquisition demand. An alphabetic character acquisition demand is performed using an alphabetic character, the character code corresponding to 1 to 1, and typeface information, in order to decide the alphabetic character to acquire. In order to acquire the demanded alphabetic data, the system control section 304 gains required data from the typeface data 301,302,303. If the gained data are profile vector data which constitutes an outline font, data will be transmitted to a rasterizer 305,306. Moreover, if the gained data are bit pattern data, it will be transmitted to the application section 307 as it is. In a rasterizer 305,306, the received profile vector data is developed to a bit pattern, and it is returned to the system control section 304. The system control section returns the bit pattern data which 304 received from the rasterizer 305,306 to the application section 307.
[0017] Next, an example is explained to a detail based on the above explanation.
[0018] This example carries out typeface data file division, aims at enabling acquisition of typeface data by installing only a required part, and is briefly explained about the division approach using drawing 4 .
[0019] Drawing 4 expresses the configuration of a division file briefly. In addition, in order to simplify explanation, Typeface A shall constitute one typeface from character codes 0-600.
[0020] This typeface A should be divided into A3 [of 0-100] of A1,101-199 of A2,200-500, and four parts of 501-600. In this typeface A, A3 part is data which generally are not used, and is a part which needs disk memory most.
[0021] The part which should show such divided typeface information at the time of typeface install, and should install it is made to choose by a user's hand (keyboard). As a result of being chosen, only the part needed is installed and the typeface information file 500 which described the items of the typeface of the installed result which is shown in drawing 5 A is created on external storage 109. In addition, a situation is shown as a result of the typeface file installed in drawing 5

B.
[0022] The information which shows the items of the typeface which read information and was installed out of the typeface information file from the typeface information file 500 at the time of a system startup is taken out, and a typeface management information table as shown in drawing 6 is created on the typeface information management field 205.
[0023] The flow of typeface management information table creation is explained using drawing 7 . In addition, this drawing is processing about the typeface under initial processing when a power source is supplied to this system, and **** procedure is stored in external storage 109.
[0024] First, a typeface information file is opened at step S702, at the following step S703, information on a typeface name is acquired from the opened typeface information file, and the information on a typeface name is written in a typeface management information table.
[0025] In step S704, a file name is acquired from a typeface information file, and the information on a file name is written in a typeface management information table. And in step S705, the file open shop operation for considering as an accessible condition is performed based on the file name of the typeface gained previously. In step S706, from the typeface file which became accessible in step S705, the information about the initiation code of the typeface data stored in the file and a termination code is read, and it writes in a typeface management information table.
[0026] When the file of the same typeface investigates whether it still exists to the description in a typeface information file and processing exists in it in step S707, acquisition of return and the following file name and an update process of a table are performed to step S704.
[0027] Moreover, when it is judged that the file of the same typeface does not exist at step S707, it progresses to step S708 and investigates whether an unsettled typeface exists in a typeface management file, and when it exists, it returns to S703 and processing to the following typeface is performed. Moreover, if it judges that the creation of a table to all typefaces was completed, it will progress to step S709, the typeface information file which performed open shop operation in previous step S702 will be closed, and this processing will be finished.
[0028] In the typeface information management field 205, a typeface management information table as shown in drawing 6 will be created by the above processing.
[0029] Next, flow until it gains the typeface data to need using the typeface management information table of drawing 6 is explained according to the flow chart of drawing 8 R> 8.
[0030] Alphabetic data income processing is started in step S801.
[0031] In step S802, the information (a character code, typeface information) about the alphabetic character acquired from the AP section 307 in drawing 3 is acquired. In the following step S803, the head of the typeface management information table of drawing 6 carries out typeface retrieval (in the initial stage, the pointer is pointing to the head typeface in a table), and it confirms whether to be a corresponding typeface. When it exists and does not exist to step S804, it progresses to step S805.
[0032] If it judges that it is not a corresponding typeface and processing progresses to step S805, it judges whether other typefaces exist, and if there are other typefaces, a pointer will be carried forward there.
[0033] Moreover, if a corresponding typeface can be discovered and processing progresses to step S804, and it puts whether the character code demanded on the typeface management information table is within the limits of [which is managed] it in another way, it will judge whether the character code is managed. Since the pointer in this time is put on the head location of drawing 6 , it will be judged whether the demanded character code is within the limits of "0" to "100."
[0034] Now, when it is judged that it does not exist, it progresses to step S806 and judges whether the file of the same typeface as the next line of the attention line in a table is managed. If it is, one pointer will be carried forward and it will return to step S804. Moreover, when it is judged that the next file does not exist, alphabetic-character-less processing is performed.
[0035] When it is judged that the character code of the demanded typeface is managed on the other hand, it progresses to step S807, and a file name is acquired from the attention line in a typeface management information table, and in step S808, typeface data are gained from the file based on the acquired file name, and it progresses to a post process S810.
[0036] While being able to stop the memory consumption of external storage 109 according to this

example since only the required character code range of a required typeface is installable in a system as explained above, since management of an unnecessary part is not performed, it also becomes possible to gather processing speed.
[0037] In the <explanation of 2nd example> above-mentioned example (the 1st example), when a typeface management information table shows drawing 6 , for example and there is an alphabetic character income demand of the character code "501" of Typeface A from the application section, the loop formation of step S804 in drawing 8 and step S806 will be carried out 3 times. To the whole demand, the frequency of the alphabetic character income demand which receives by "600" from a character code "501" is satisfactory, in not being so high, but to the character code of others [frequency / the], when many, the fall of processing speed is not avoided.
[0038] Then, you carry out counting of the occurrence frequency of an alphabetic character income demand, and make it reflected in the typeface management information table creation at the time of next starting in the example of **** 2.
[0039] In addition, the system configuration in the example of **** 2 shall also be shown in drawing 1 . This is the same about the 3rd example or subsequent ones mentioned later. Moreover, in the example of **** 2, when installing typeface data in external storage 109, the case where the typeface data A1 of all the range shown in drawing 4 - A4 are specified is explained. Moreover, the program concerning the flow chart of drawing 8 is memorized by external storage 109 with the natural thing. The same is said of each example which also mentions this point later.
[0040] Drawing 9 shows the initial state of the typeface managed table created in the 2nd example. The difference from drawing 6 of the 1st example is the point that the column which carries out storage maintenance of the count of access of each code range in the same typeface was prepared. Since the procedure which creates the typeface management information table itself is the same as that of the 1st example, explanation here is omitted and explains alphabetic data income processing according to drawing 10 .
[0041] The difference between the flow chart of drawing 10 and drawing 8 in the 1st example is a point that step S811 was added by step S807 becoming step S807'.
[0042] In illustration, when a typeface managed table has an alphabetic data income demand from the application section and the demanded alphabetic data is managed on the managed table, as for processing, it is the same to progress to step S807' from step S804. If processing progresses to step S807, while acquiring a file name from a typeface management information table, "1" ink MENTO of the numeric value of the access column to which a typeface management information table corresponds is carried out. And in step S808, based on the acquired file name, typeface data are gained from the file and it progresses to a post process S811.
[0043] As a result, for example, a typeface management information table, it comes to be shown in drawing 11 .
[0044] At step S811, processing which puts in order and changes the sequence memorized in the typeface management file by the count of access in the managed table in the time is performed, and this processing is finished.
[0045] For example, about Typeface A, when a typeface management information table suits the condition which shows in drawing 11 , since the count of access of a file is A4>A3>A2>A1, as the contents of the typeface information file are shown in drawing 12 , the sequence is changed.
[0046] Since a table is created according to the sequence memorized by the typeface information file at the time of equipment starting, the typeface management information table at the time of next starting comes to be shown in drawing 13 . that is, file A4 in Typeface A is located in the very first — ******** — alphabetic data — improvement of an access rate to a profit demand can be desired in some places.
[0047] In addition, although the typeface information file was updated in the 2nd example whenever the alphabetic data income demand occurred, it becomes possible by carrying out at the time of power off processing of a system to accelerate processing speed more.
[0048] moreover -- as a file also with the suitable count of access of the partial file (A1 - A4) which constitutes each typeface in system power off processing -- saving -- the time of a next system startup — the file to the count of access -- income -- carrying out — still — **** -- you may make it determine the sequence in a table

[0049] It becomes possible to raise the cost performance of the whole which starts character-pattern generating since priority is given to the character code range with many counts of access and it arranges at the head of a typeface information management table according to the example of **** 2 like explained above.
[0050] <Explanation of the 3rd example>, next the 3rd example are explained. In the example of **** 3, the code range which the user specified among two or more typefaces is replaced with other typefaces.
[0051] Drawing 14 shows the configuration of the division file in the example of **** 3. A different point from it of drawing 4 R> 4 in the 1st previous example is that two or more typefaces are divided. In addition, in illustration, A1 - A3, B1-B3, and C2 show the file name, respectively.
[0052] In the example of **** 3, from a file C2 (in fact the typeface C), data will be taken out and the range of character code "101"- "199" in Typeface A will be processed, for example, if it of Typeface C is subordinated, for example, the demand of the alphabetic data of the character code 101 of Typeface A comes from application.
[0053] According to this, while the alphabet, a figure, the Taira kana, katakana, the kanji, etc. use one a "typeface name", it becomes possible to generate the pattern of the typeface considered as a request for every classification of the above-mentioned alphabetic character, for example. Moreover, the whole typeface A is an outline font, for example, and when Typeface C is a dot pattern, it becomes possible to process a specific part at a high speed depending on the case.
[0054] Now, the example of contents of the typeface information file in the example of **** 3 is shown in drawing 15 . A user may create creation of this typeface information file freely, and a system may create it automatically. Or a user may be made to select typeface data in the phase installed in this system, as the 1st example of the above showed.
[0055] If the part of the file A2 of Typeface A is subordinated to the part to which Typeface C corresponds as shown in drawing 15 , since installing in external storage 109 becomes unnecessary, the file A2 in Typeface A can also stop the memory consumption of external storage 109. Furthermore, since it has only to determine which range of a character code is used with which typeface also when a user registers his own typeface, the amount of memory substantially consumed with a user typeface becomes only the typeface information file which showed the dependency. In addition, if the information which shows whether it is that it is subordinate is added and each typeface has some which are subordinate to the range in whether it is subordinate for every character code range with one typeface as shown in drawing 15 , the information the typeface indicates something to be is added.
[0056] Now, if this system starts when there is a typeface information file as shown in drawing 15 , a typeface management information table as shown in drawing 16 will be created on the typeface information management field of RAM103.
[0057] Processing concerning creation of this typeface management information table is explained according to the flow chart of drawing 17 .
[0058] If this processing is started in step S751, in order to create a typeface management information table, the field about ** will be secured in the typeface information management field 205. And it progresses to step S752, the typeface information file saved at external storage 109 is opened, and typeface information is acquired from the opened typeface information file at the following step S753.
[0059] If it judges that they are not return and a subordination typeface to step S753 when processing progresses to step S754 and the typeface currently observed is the subordination processing which judges whether it is a subordination typeface, it will progress to step S755.
[0060] That processing progresses to step S755 means the table creation initiation to one typeface. Therefore, the typeface name is written in a typeface information management table, and the table which clarified the dependency for every character code range to the typeface by the processing explained below is built here.
[0061] At step S756, the following information, i.e., the subordination information over the code range in the typeface currently observed, is taken out from the opened typeface information file. And at step S757, it judges whether the taken-out character code range has subordination information in the taken-out information, if it puts whether it is that it is subordinate in another

way.
[0062] If there is no subordination information, it will write in a typeface information management table so that it may correspond to the code range which is observing the file name of a proper at the typeface which is progressing and observing step S758.
[0063] Moreover, if there is subordination information, it will progress to step S759, the file name to which the subordinate typeface corresponds will be taken out, and it will be written in a typeface information management table.
[0064] Processing progresses to step S760, based on the file name acquired step S758 or 759, opens the file and, in any case, is made accessible. And in the following step S761, it becomes accessible, and from a typeface file, the information about the initiation code and termination code of typeface data which are stored in the file is read, and it is written in a typeface information management table.
[0065] If processing progresses to step S762, and it judges that it judges whether the processing to one typeface was completed, and has not ended, it will return to step S756 and the above-mentioned processing will be performed.
[0066] Moreover, when it is judged that the processing to all the code range to one typeface was completed, it progresses to step S763 and judges whether there is any typeface which should be processed next. If the typeface which should be processed remains, processing will progress to step S764, if there are not return and a typeface which should be processed in step S753.
[0067] When processing progresses to step S764, the opened typeface information file is closed and this processing is ended at step S765.
[0068] When there is a typeface information file as shown in the above result, for example, drawing 15 , a typeface information management table as shown in drawing 16 will be created.
[0069] In addition, the processing when gaining typeface data using the typeface information management table created in this way is the same as that of the 1st example explained previously. That is, if there is a typeface data capture demand of the character code "101" of Typeface A from the application section, since it will investigate the table and it will be detected that the character code "101" to Typeface A is a file name C2, the typeface data which correspond from the file C2 are taken out, and a character pattern is generated. When this typeface data is outline data, a rasterizer 305 is once passed, a dot pattern will be generated and it will be passed to application.
[0070] Even if it does not change a typeface one by one when creating the text with which English and Japanese were mixed by replacing the data of another typeface in the code range considered as the request of one certain typeface according to the example of **** 3 as explained above, the alphabetic character input in the typeface by which a user's volition was reflected in each can be performed.
[0071] and the time of install -- or since the typeface data which were in the code range to which it pointed so that it might be subordinate to the time amount of arbitration in the data of other typefaces from the first become unnecessary substantially, and the part is removed for install or it becomes possible to delete, it becomes possible to stop the memory consumption of external storage 109.
[0072] The 4th example of <explanation of the 4th example> is explained below. Usually, when installing typeface data in a system, it is indispensable to memorize the one whole typeface to one logic storage device. The example of **** 4 makes it possible to memorize one predetermined storage place to a logically different device. The demand of wanting to prepare the amount of ways as for which does not have only the availability for example, each logical device remembers all one typefaces to be, or a certain extent is vacant in each logical device with this is met.
[0073] In addition, the logic storage device said here is a concept also containing a physical unit, for example, when one physical storage is logically divided as two or more logical equipments, or even if it is the same physical unit, it is a concept also containing that from which pass differs.
[0074] The example of the typeface information file in the example of **** 4 is shown in drawing 18 . In addition, by illustration, in order to simplify explanation, it was shown only to one typeface. In illustration, A1 - A4 are the file names of the typeface A corresponding to the code range, as the above 1st - the 3rd example explained.

[0075] If illustration is explained briefly, it is shown that the typeface data of a certain code range to Typeface A are memorized by logical drive A as a file A1.
[0076] Now, in the example of **** 4, this typeface information table is taken out at the time of a system startup, and a typeface information management table as shown in drawing 19 based on the data described in it is created.
[0077] It differs in that the drive name of a storage place is indicated to the format which showed the format of the typeface information management table created based on the typeface information file and it in the 4th example to drawing 5 [of the 1st example] A, and drawing 6 . If between sees in the structure of a typeface information file shown in the contents of processing and drawing 18 which the 1st example explained previously will require if it is this contractor, it will guess easily. Therefore, the explanation is omitted here.
[0078] However, although it explained in the example of **** 4 as what directs the storage place of the file of each code range which constitutes typeface data when installing, of course, the same storage drive may be made to once memorize all, and the typeface data of the suitable code range may be moved to another storage (for example, more nearly high-speed external storage) after that if needed.
[0079] Since it is memorizable to the device which according to the example of **** 4 there does not need to be a typeface data in the same location of one logical device, and is considered as a request as explained above, it becomes possible to utilize external storage effectively.
[0080] <Explanation of the 5th example>, next the 5th example are explained. For example, even if it is the case where the design of the alphabetic character of a certain specific range is used, since it is necessary to memorize the typeface data of all the range of the typeface of the design, memory consumption, such as external storage, cannot be disregarded. This technical problem is swept away in the example of **** 5.
[0081] Drawing 20 shows the configuration of a division file briefly. In illustration, Typeface A shows that the one whole alphabetic character typeface is constituted from character codes 0-500, and has managed it as code range of A3 of A2,200-500 partial [three] of A1,101-199 of 0-100. Typeface A' is the same as Typeface A, corresponding character sets differ, it has only a part for the alphabetic character data division which must be changed at the time of the change of character sets, and the character code contained is the same as that of 101 to 199A2. That is, as a typeface, although A' is the same as A, the character codes to be used are only 101-199.
[0082] In addition, in the same code range (space), character expression is said to condition, such as the Taira kana, the alphabet of katakana and English, and the Russian alphabet, as character sets.
[0083] In order to perform the above-mentioned typeface management, as the typeface information file in the example of **** 5 is shown in drawing 21 , the relation of a typeface file and character sets is described.
[0084] Since the creation procedure of the typeface management information table built to the typeface management information field 205 in RAM103 at the time of a system startup is performed by the procedure shown in drawing 7 , explanation here is omitted. Consequently, the typeface management information table shown in drawing 22 is built. A1, A2, A3, and A2' are the file names of the data which build the typeface data A and A' among a table, respectively.
[0085] In addition, although performed by the procedure which also showed the flow of alphabetic data income to drawing 8 using the typeface management information table created by doing in this way, it will be as follows if it explains.
[0086] First, alphabetic data income processing is started in step S801.
[0087] In step S802, the information (a character code, typeface information, character sets) about the alphabetic character acquired from the AP section 307 in drawing 3 is acquired. In the following step S803, it is confirmed whether search the typeface management information table of drawing 6 , and a corresponding typeface and character sets exist. When it exists and does not exist to step S804, it progresses to step S805.
[0088] At step S805, it judges whether another typeface still exists on the typeface management information table shown in drawing 22 . In existing, it progresses to step S803, and the check to another typeface in a table is performed.

[0089] Moreover, if it judges that the demanded typeface exists in a table and progresses to step S804, it will judge whether the code demanded in the initiation code of the attention character sets in an attention typeface and the termination code is contained. In addition, since "-1" shall be stored for the initiation code and the termination code about the file not existing, the code range can judge easily whether the alphabetic data of the code is managed.
[0090] Now, in step S804, when it is judged that it is out of range [the character sets which the demanded character code is observing], it judges whether it progresses to step S806 and there are any character sets which are not yet checked in the same typeface. If it is, it will judge [return and] whether the character sets again corresponding to a demand code exist to step S804. Moreover, when there is nothing, it progresses to step S809 and alphabetic-character-less processing is performed.
[0091] When it is judged that the character sets which have, on the other hand, managed the typeface and character code of the demanded alphabetic character exist, it progresses to step S807, a file name is acquired from a typeface management information table, and income of the alphabetic data which corresponds from the file of the file name at step S806 is carried out. And it progresses to step S810 and processing is ended.
[0092] As explained above, even if it is the case where the alphabetic character of character sets which are different even if it is the same typeface is specified according to the example of **** 5, that what is necessary is just to have typeface data of the code range only containing the alphabetic character to be used, the data corresponding to a whole sentence character code become unnecessary, and can utilize the storage region of external storage effectively.
[0093] Though the same typeface A' as Typeface A is used, for example, when using different character sets in the 5th example of the <explanation of 6th example> above, it was what aims at a deployment of memory by storing only the typeface data of the character code range to be used in external storage.
[0094] It is the example which memorizes a different typeface in the example of **** 6, and utilizes the typeface data of the same range of the typeface of another side as typeface data of the specific code range in one typeface. Consequently, the need of overlapping and having typeface data of the part used in common is abolished, and efficient use of memory is enabled.
[0095] Drawing 23 shows briefly the configuration of the division file in the example of **** 6. In illustration, Typeface A shows that the one whole alphabetic character typeface is constituted from character codes 0-500, and has managed it as code range of A3 of A2,200-500 partial [three] of A1,101-199 of 0-100.
[0096] Moreover, like Typeface A, three are divided, whole sentence character code space is managed, among these Typeface B uses typeface data peculiar to Typeface B, and, as for character codes 0-100, and 200-500, uses the thing of Typeface A about codes 101-199.
[0097] In order to perform the above-mentioned typeface management, the typeface information file in the example of **** 5 consists of division typeface data file names which build the class and typeface of a typeface, as shown in drawing 24 .
[0098] Although explanation here is omitted since the creation procedure of the typeface management information table built to the typeface management information field 205 in RAM103 at the time of a system startup is performed by the procedure shown in drawing 7 , the typeface management information table shown anyway in drawing 25 is built all over the typeface information management field 205 in RAM103.
[0099] What is necessary is just to follow the flow chart which also showed alphabetic data income processing to drawing 8 . That is, when there is an alphabetic data income demand from the AP section, it judges which typeface of a typeface information management table was chosen based on the typeface data specified in the demand, income of the file name which has managed the character code as which it was specified in the typeface next is carried out, and income of the alphabetic data which corresponds from the file is carried out after that.
[0100] since the typeface name showing a typeface or a typeface number, and ** were distinguished using separately a typeface name and a typeface number or the style information by the need, width-of-face information, and weight information when typeface data were distinguished until now [<explanation of 7th example>] -- the distinction processing -- complicated -- not

becoming -- it did not obtain but there was a problem of taking time amount.
[0101] So, the example of **** 7 explains the example which makes it possible to perform this distinction at a high speed simply.
[0102] Drawing 26 shows the format of the style width-of-face weight information in the example of **** 7. Like illustration, 8 bits (0-255) are assigned as 4 bits and weight information as style information as 4 bits (0-15 as a value), and width-of-face information. As style information, "0" defined the definition and regular and "1" were defined as italic and --. Moreover, 4, 6, 8, 10, and 12 are made to correspond to extra condence, condence, regular, expand, and extra expand as width-of-face information. And 60, 70, 80, 90, 100, and 110,120,130,140 were made to correspond to inside **, **, super-thick, and **** into ****, super-thin, **, and inside ** as weight information.
[0103] Now, the installation procedure (install to a hard disk from the floppy disk of external storage 109) in this example is explained according to the flow chart of drawing 27 .
[0104] First, it requires that the media in which the typeface data to install and data (install data file) required for install are stored should be set up, and the media are made to specify it as a user in step S301.
[0105] Next, typeface name information, a typeface number, and style weight information are read from the install data file in which the typeface data progressed and installed in step S302 are stored.
[0106] At step S303, the typeface data in the typeface information file already installed in the install place are read. If it judges that the typeface information file which should be read does not exist in an install place at this time at step S304, that typeface file will be created (step S305).
[0107] Next, processing progresses to step S306 and compares the contents of the typeface information file of an install place with the typeface information which it is going to install. Priority of comparison processing is made into a typeface number the 1st, and makes it style width-of-face weight information and the 3rd typeface name the 2nd. All the above items are the same typefaces unconditionally, when the same. Moreover, when the contents of the 1st item and the 2nd item are the same, since the alphabetic data of a typeface is the same, it decides [the processing with which it is made for the contents of the typeface not to lap and] by this processing whether it processes uniquely including a typeface name (for example, it is made to direct to a user).
[0108] Now, if processing progresses to step S307, it will judge whether the typeface data which it is going to install by this install processing are updating or a new addition. As the update process was explained previously, all of three items are the same cases.
[0109] When it is judged that it is an update process, it progresses to step S309, and in order to overwrite the already installed typeface data, a same typeface's described by typeface information file's existence location is overwritten. If the install place is separately directed by the user at this time, processing written in that directed location will be performed.
[0110] Moreover, when it is judged that it is new typeface install, it progresses to step S308 and a typeface name, a typeface number, style width-of-face weight information, a tag file name, and all typeface information are installed.
[0111] And finally this processing is ended and it returns to a calling agency.
[0112] Drawing 28 shows the contents of install media, and the contents of the install data file in it.

[0113] Information required for install is described by the install data file. Typeface information is stored in the typeface file. The data for creating a typeface from the information on a typeface file are stored in the tag file. Therefore, it is thought that the typeface for a tag file exists using two or more tag files from one typeface data.
[0114] Drawing 29 shows an example of the contents of the typeface information file in this example. "NAME(s)" are a disk RIPUSHON name and typeface information (copy the main typeface number, a body number, style width-of-face weight information). "DEFTAG(s)" are a tag file name and a typeface name which exists in a tag file. Moreover, "FILE(s)" are the install place of a typeface disk, and the file name of a typeface file.
[0115] Drawing 30 shows the example of contents of a tag file. A FEISU name is stored in the size

of a tag file, a typeface number, style width-of-face weight information, a disk RIPUSHON name, and typeface metrics information at this file. Related **** to typeface data is performed from the typeface number in this, and style wait information.
[0116] Drawing 31 shows the contents of the typeface file. Typeface configuration information (file size), typeface data (a typeface initiation code, the number of alphabetic characters), a typeface number, and style width-of-face wait information are stored in this file.
[0117] Like the above, according to the example of **** 7, after install is having the style width-of-face wait which doubled the style information on typeface information, width-of-face information, and wait information as information showing a typeface, and distinction of a typeface is simply attained at a high speed rather than it compares a typeface name, a typeface number, etc.
[0118] The 8th example of <explanation of the 8th example> is explained. The example of **** 8 explains the principle which makes a font-related file name optimize. The file name which the system (or OS) in this example manages as an example explains the example the number of eight characters (8 bytes) and extensions is [example] three (3 bytes).
[0119] Drawing 32 shows the alphabetic character structure of using for the file name of the typeface physical file in the example of **** 8.
[0120] In illustration, one figure as which two characters for typeface discernment which show ** to drawing 34 are located, and express to ** the style width-of-face information (0-9) shown by drawing 36 is located. ** **** -- it is the size support field information which the figure showing the wait information shown in drawing 37 is located, and is shown in ** by drawing 38 . ** It is file identification information and use a character string "JFD" in the example.
[0121] Moreover, drawing 33 shows the structure of the file name of the typeface logic data file of typeface data. ** It is typeface management information and express using three characters of arbitration. ** The sequence of a release of this file is shown and apply by drawing 35 . ** It is the wait information shown in drawing 37 . ** metrics information identification information -- such -- **. this paragraph eye shows the sequence of a release of this file. Moreover, when it has the same information as a printer font, it expresses with "P." ** It is file identification information and use "FON" in this case.
[0122] Hereafter, it explains in more detail about drawing 38 from drawing 34 .
[0123] Drawing 34 is the identification information of a physical typeface. In a block letter and RG, a round-head block letter and KA express a square style object, and KY expresses [MI / a Mincho typeface and GO] a textbook object. In a Mincho typeface and GOT, a round-head block letter and KAI express a square style object, and, as for KYO, a block letter and RGO express [MIN in drawing 35] the textbook object, respectively.
[0124] Drawing 36 shows style width-of-face information. As a style, there are two kinds, regular and italic, and there are extra condence, condence, regular, expand, and extra expand as width-of-face information. In regular and extra condence, with this combination, "1" was assigned like illustration. Similarly, 4 was assigned to "3", regular, and extraexpand to regular and condence at "2", regular, and expand at "1", regular, and regular. Moreover, in italicr and extra condence, "5" was assigned and "9" was assigned to "8", italic, and extra expand to italic and condence at "7", italic, and expand at "6", italic, and regular.
[0125] Drawing 37 showed wait information and assigned figures 1-9 in the example to inside **, **, super-thick, and **** into ****, super-thin, **, and inside **.
[0126] Drawing 38 is the size support field information in an example. Size data are expressed with a hexadecimal using 8 bits of high orders of these data (when it says in written form, it is "F" from "1"). In addition, in being scalable typeface data, such as outline data, it uses a figure "0." Moreover, in being the field which assigns each field to 8 bits of low order, and corresponds to it, it sets the bit to "1." an example -- a bit 7 -- a non-kanji character field and a bit 6 -- the 1st level field and a bit 5 -- the appropriate field and the bit 1 were made into the external-character field, and the bit 0 was made [the 2nd level field and the bit 4 / the old JIS field and the bit 3] into the special field for the kana field and the bit 2.
[0127] The above determines a file name. Hereafter, the example of decision using typeface information and its typeface information is shown.
[0128] It regular(s). typeface information [1. -- typeface name: -- Mincho typeface and 2. style

name: -- 3. -- character width information: -- inside [of condence and 4. alphabetic character wait information:], and 5. character-size: -- 100 dots 6. Since it is a support field, the 1st level, and the same metrics information as a 7. metrics information identification information:printer font, "P", The file classification by eight extensions: JFD (typeface physics data file), FON (typeface logic data file), and 9:typefaces management environmental name:FontManaging} determine.
[0129] In this case, in order to determine the name of a typeface physics data file, ** shown by drawing 32 is set to "JFD." ** It is set to "MI" by drawing 34 . Moreover, by drawing 36 , ** is set to "1" by drawing 37 , and ** is set to "5." When size 100 is expressed with a hexadecimal, it is 64H, since only a bit 6 is stood according to drawing 38 R> 8, 8 bits of low order are set to 40H therefore, and ** is set to "6440."
[0130] That is, it is a file name "MI156440.JFD."
[0131] Next, since it unifies into in the case of a typeface logic data file extension ** expressing "FON" and ** expressing FontManager, "FO_ and ** are ["5" and ** of "MIN" and **] "P" by drawing 37 from drawing 35 .
[0132] Therefore, the name of this typeface logic data file is "FM_MIN5P.FON."
[0133] As explained above, according to the example of **** 8, the information stored in the typeface file can make it intelligible by having displayed the information which shows the data stored in the file on the typeface file name.
[0134] The 9th example of <explanation of the 9th example> is explained.
[0135] Usually, it remembers beforehand that the bit map pattern explained previously for character-pattern generating, and there are a bit-mapped font method which carries out income of it, and an outline font method which obtains a character pattern by forming the profile of an alphabetic character based on outline data, and filling the interior in it.
[0136] Since the former will end without being an effective method and crushing an alphabetic character to a comparatively small alphabetic character (small alphabetic character of a dot configuration) if it is operated on a curtailed schedule to a basic character pattern even if it generates it, it can be processed at a high speed. Although a rate falls since the latter needs the processing which draws a profile and fills the interior with a rasterizer, the grace of a large alphabetic character (large alphabetic character of a final dot configuration) can be raised.
[0137] However, by the bit-mapped font method suitable for generating a comparatively small alphabetic character, from the basic character pattern, the character pattern demanded by the comparatively high probability and the character pattern of a low probability are performed as uniform processing is also, and the room of an improvement still remains.
[0138] The generating method by the bit-mapped font is made to improve further in the example of **** 9.
[0139] Now, the example of **** 9 explains the example which the resource of the bit-mapped font of the dot of 20x20 is carried out, and generates the character pattern of 4x4 to 20x20.
[0140] Drawing 39 is the flow chart which showed the procedure of the character-pattern generating processing by the bit-mapped font in an example.
[0141] First, income of the code of the demanded alphabetic character and the size of an alphabetic character is carried out at step S1. And in step S2, the character pattern (20x20 dots) of the demanded character code is taken out from ROM102 or external storage 109, and it develops in the predetermined area in RAM103.
[0142] the ** which is not deformed by one processing at step S3 -- the number of Rhine of a reducible lengthwise direction -- computing . In below one half of the character size (20x20) which the demanded character size gained previously, in being other to the one half of a character pattern, it asks by the (character size +2 before contraction) / 5. This formula is a formula for the purpose of preventing deformation (bias) by making 1 time of the number of Rhine to reduce into four lines or less, when it limits to the character pattern of 20x20 or less dots.
[0143] If processing progresses to step S4, actual contraction processing will be performed. Here, contraction processing for Rhine called for at step S3 is performed. At this time, the location which performs contraction processing is a division location, such as having asked by width-of-face/ (+one contraction Rhine) of a character pattern. Since a denominator becomes odd number, at the time of contraction Rhine number =2, it is right, and it cannot carry out a division-into-equal-parts

rate, but may be deformed here (refer to drawing 40). Therefore, amendment processing only whose remaining parts to have generated from count of said contraction location at the time of the 2nd contraction shift a location is performed. Drawing 4141 showed this. Moreover, the number of contraction Rhine = in 4, although a denominator serves as odd number, since it does not become a problem when the character pattern before contraction is 20x20 or less dots, especially processing is not performed. In Rhine of the computed lengthwise direction, the dot pattern of one line is generated by two lines of Rhine of the right-hand, the location which carries out superposition (OR) processing, and Rhine of right-hand of Rhine which will be reduced if it ****.
[0144] At step S5, it judges by judging whether the reduced result became the demanded size by the difference of the total number of contraction Rhine, and the number of Rhine reduced by step S4. When not fulfilling the demanded size as a result of a judgment, return and contraction processing will be repeated to step S3. Moreover, when creation of the demanded character pattern is completed, the character pattern generated in the requiring agency at step S6 is outputted.
[0145] In addition, the semantics of holding down the number of Rhine in one contraction processing to reduce to four or less by the above-mentioned explanation For example, when generating the pattern of 12x12 from the basic size of 20x20 (eight lines (= 20-12) are finally reduced), in the 1st time, all are not reduced by eight lines. It means reducing four lines by the 1st time, and reducing four more lines after that. As explained previously, when reducing by two lines incidentally (i.e., when generating the character pattern of 18 lines), since the number of contraction Rhine is four or less, it will be satisfactory. However, amendment processing explained previously is performed.
[0146] Moreover, although the example of **** 9 explained the example reduced to a longitudinal direction, since the contraction to a vertical lengthwise direction can completely be processed similarly and every direction can also process it further, this invention is not limited from the above-mentioned contents.
[0147] Moreover, the bit map to be used is not limited to 20x20, and if it can be efficiently processed in case it reduces in each size, it will not be limited to especially one.
[0148] Moreover, two or more bit-mapped fonts of the minimum size which can carry out income beforehand by one processing of 20x20, 16x16, 13x13, and 10x10 grade of "/(character size +2 before contraction) 5" may be prepared, and the number of loop formations of steps S3-S5 may be reduced, and you may use for improvement in grace.
[0149] Character-pattern contraction with sufficient performance can be performed by only the high size (4x4 to 20x20) of possibility that the size of the character pattern which was demanded according to the example of **** 9 will be most used by this creation etc. using a bit-mapped font, as explained above, and being simplified so that it may become the most effective in the size which had count of the location which contracts limited.
[0150] The example of <explanation of 10th example> **** 10 is explained.
[0151] The file made data with a file name reflect in the above-mentioned example (for example, the 8th example). Therefore, what is necessary is just to manage so that a different file name may be attached when the side which offers a typeface newly offers a different name from the name supplied in the past. However, when the user of a system enables it to create a typeface, the file name which the user is using cannot be managed in a supply side. Therefore, it must copy, after copying the typeface created after deleting the typeface which the typeface which has the same information may be created and once includes the same information or changing the typeface information to which the same information corresponds.
[0152] This problem is solved in the example of **** 10.
[0153] Drawing 42 shows an example of the file name attached to the typeface information file in an example. Like illustration, three characters (3 bytes) are used for the part of a file name in the example of **** 10 at eight characters (8 bytes) and an extension. In this example, "FG_US" considers as the thing of immobilization and the same value as a typeface number is attached to ** shown after that. However, how to attach a file name is not restricted to this.
[0154] Drawing 43 shows the internal structure of a typeface information file. In this typeface information file, the typeface name shown with the sign 14 of illustration and the typeface number

shown with a sign 15 are stored. Moreover, information, such as the date and time of creation and a version, is also stored in others.

[0155] Drawing 44 shows the structure of the typeface storing information file which manages the storing place of the typeface memorized by external storage 109, and the information about the complete-works object memorized is contained. They are the typeface information file name of the typeface 1 of the registration number of a typeface 1 shown in this typeface storing information file with a sign 16, and a sign 17, the typeface name of the typeface 1 of 18, and the typeface number of the typeface 1 of 19. the same -- a typeface 2 and -- Typeface n -- it is alike, respectively, and it also receives and the same information is stored.

[0156] Based on the flow chart of drawing 45 , the typeface management procedure in the example of **** 10 is explained.

[0157] First, in step S501, income of the typeface information is carried out from the typeface information file (the typeface information file which has managed the typeface data which the user created, 43 reference) of a copied material. Next, it progresses to step S502 and income of the typeface information about the typeface stored from the storing information file (R> drawing 44 4 reference) of the typeface put under management of a system is carried out.

[0158] In step S503, it judges whether the typeface of the same typeface name is stored from the information which carried out income at previous steps S501 and S502. When it is judged that the typeface of the typeface name is not registered into step S504 by the system when it is judged that the typeface of the same typeface name is already stored in a typeface storing information file, it progresses to step S507.

[0159] If the same typeface name is judged not to register with a system and progresses to step S507 here, it will judge whether the same typeface number as the typeface number of the typeface which the user created is registered into the system. If it judges that there is no same typeface also here, processing will progress to steps S512 and S513, will copy the contents of the typeface information file which the user created to the typeface storing information file which has memorized the typeface information registered into the system (addition), and will update the typeface storing information file.

[0160] On the other hand, at step S503, when it is judged that the typeface of the same typeface name is already registered, it progresses and swerves to step S504, and it judges whether ** is the same typeface number.

[0161] When it was the same typeface number, i.e., it is judged that both the typeface name and the typeface number have already registered with the system, it progresses to steps S505 and S506, the location where it corresponds in a typeface storing information file is overwritten from the contents of the typeface information file which the user created, and a typeface storing information file is updated.

[0162] When it is judged at step S507 that it is "YES" (i.e., although it is not the same typeface name, when it judges that it is the same typeface number), it progresses to step S508, and the typeface number for which the typeface which it is newly going to register is not used is detected, and it progresses to step S509.

[0163] Also when decision of step S504 judges that typeface numbers differ although "NO, i.e., the same typeface name," exists, processing progresses to step S509. In this case, a typeface number will overlap.

[0164] Anyway, that processing comes to step S509 has a the same typeface name, and it is the case where typeface numbers differ. It sees from a user side, and it will become the base of derangement, if a typeface name is important and there are two or more same typeface names. So, at steps S509, S510, and S511, a typeface storing information file is updated by changing the file of the typeface name which it is going to register (for example, "USR" being added to a typeface name), and adding the changed typeface name to a typeface storing information file, and registering by the typeface number.

[0165] In the above-mentioned example, it becomes possible from distinguishing the registration condition of a typeface by making a typeface name into key information, making a typeface number correspond with the file name of a typeface information file, and copying by changing a typeface number to manage the typeface in which the same typeface information is included. Moreover, this

invention offers the tool which can create a typeface to a user, and when using the created typeface by said tool, it demonstrates great effectiveness.
[0166] In addition, although a typeface name is used as a key, typeface information is retrieved in the above-mentioned example and the typeface number was changed in it, the key information and the typeface information to change which use a typeface number as a key, and search it, and you may make it change a typeface name, and are used for retrieval are not this limitation.
[0167] By having made typeface information and the file name of a typeface information file correspond, and having been made to copy according to the example of **** 10 by changing typeface information like, when the typeface which was explained above and in which the same typeface information is included was detected (registration), even if it is the case where the same typeface information is included, it becomes possible to manage a typeface.
[0168] Moreover, it becomes possible to aim at mitigation of the activity which copies the typeface created after deleting the typeface which once includes the same information even when the typeface which has the same information when a user can create a typeface is created, or copies after changing the typeface information applicable to the same information.
[0169] Even if it applies the above 1st – not only the 10th example but this invention to the system which consists of two or more devices, it may be applied to the equipment which consists of one device. Moreover, it cannot be overemphasized that this invention can be applied also when attained by supplying a program to a system or equipment.
[0170]
[Effect of the Invention] As explained above, according to this invention, it becomes possible to offer the management method of supple typeface data.
[0171]

[Translation done.]

TECHNICAL FIELD

[Industrial Application] This invention relates to the generating approach of the character pattern generated as the typeface data control approach and it which manage the data file of the various typefaces used with the typeface data control approach and the character-pattern generating approach, for example, a personal computer, a word processor, desktop-publishing equipment, etc. are also.

[Translation done.]

PRIOR ART

[Description of the Prior Art] Conventionally, the typeface data managed in a character-manipulation system stored the data of a whole sentence character per typeface in one file from the reasons of the ease of management etc.

[Translation done.]

EFFECT OF THE INVENTION

[Effect of the Invention] As explained above, according to this invention, it becomes possible to offer the management method of supple typeface data.

[Translation done.]

TECHNICAL PROBLEM

[Problem(s) to be Solved by the Invention] There is a problem shown below by the typeface data control approach by the above-mentioned conventional approach.
[0004] All the typeface data are beforehand stored in the file, and the object of install had only selection in a typeface unit to the last. For this reason, even the typeface data which do not not much have being used generally will be installed in coincidence, and it had resulted in pressing disk memory.

[Translation done.]

MEANS

[Means for Solving the Problem] This invention tends to be accomplished in view of this conventional technique, and it is going to offer the management method of supple typeface data.
[0006] For this reason, the one typeface data control approach of this invention is equipped with the stroke shown below, for example. That is, in the typeface data control approach for taking out the data about the demanded alphabetic character from a typeface data file, the typeface data file which divided character code space into two or more partial code space, and became independent for every divided character code range is managed.

OPERATION

[Function] In the configuration in the configuration of this this invention, one typeface data is divided into plurality in character code space, and, moreover, has the file of typeface data for every divided partial code space of the.

[Translation done.]

EXAMPLE

[Example] Hereafter, the invention in this application is explained to a detail with reference to a drawing.
[0009] First, in case the system treating alphabetic data, such as a personal computer, performs a display or printing of an alphabetic character, a motion until it gains typeface data is explained briefly.
[0010] Drawing 1 is the block diagram showing a fundamental configuration for the system of these above to operate.
[0011] In drawing 1 , 101 is CPU, i.e., a central processing unit, and performs starting of the above-mentioned whole system, data processing, etc. 102 is a read-only memory (ROM) and is storage regions, such as a system bootstrap (boot program) and character-pattern data. 103 is random access memory (RAM), and it is used for storage temporarily [various data] while a system program, various application programs, the driver program for character-pattern generating, etc. are loaded. Of course, the program concerning the flow chart mentioned later is also loaded to this RAM103, and is performed. 104 is a keyboard control section (KBC) and transmits key input data to reception and CPU101 from the keyboard (KB) of 105. 106 is a CRT control section (CRTC) which controls a display unit (CRT) 107. 109 is external storage, such as a floppy disk drive unit (FD) and a hard disk drive unit (HD), and various programs and font data including a system program or an application program are stored in these. The program memorized by these external storage 109 will be loaded to RAM103, and will be performed by CPU101 at the time of program execution. 110 is a printer control section (PRTC) and 111 is a printer (PRT) which forms the visible image based on the data sent out under control of PRTC110 in the record paper. 112 is a system bus and should become the path of the data between above-mentioned components.
[0012] Drawing 2 is drawing having shown the memory map of RAM103. 201 is a system area where a system is loaded, 202 is a field where the alphabetic character expansion processing section is loaded, and 203 is a field where the alphabetic character expansion processing section stores bit pattern data. 204 is a field read in order to refer to the vector pattern data stored on the external storage of 109 at the time of alphabetic character expansion processing. 205 is a field for generating a management information table for said system operating. 206 is the cache field of typeface vector data.
[0013] Drawing 3 is drawing showing the structure of a system treating said alphabetic data.
[0014] In drawing 3 , 301,302,303 expresses various typeface data memorized in the external storage region 109 in drawing 1 . A limit does not have substantially the class of typeface data memorized by external storage 109, and its number.
[0015] 304 shows the system control section and 201,204,205 in drawing 2 is contained. The alphabetic character expansion processing section, i.e., a rasterizer, is shown, only software may realize the alphabetic character expansion processing facility, hardware may be realized, and even if 305,306 uses the both sides of hardware and software, it is not cared about. Moreover, if at least one number of the rasterizers contained in said system is contained, there will be no other limits. 307 expresses the application section (for example, program of a word processor etc.) which publishes an alphabetic character acquisition demand to a system.
[0016] In drawing 3 , the application section 307 emits an alphabetic character acquisition demand.

An alphabetic character acquisition demand is performed using an alphabetic character, the character code corresponding to 1 to 1, and typeface information, in order to decide the alphabetic character to acquire. In order to acquire the demanded alphabetic data, the system control section 304 gains required data from the typeface data 301,302,303. If the gained data are profile vector data which constitutes an outline font, data will be transmitted to a rasterizer 305,306. Moreover, if the gained data are bit pattern data, it will be transmitted to the application section 307 as it is. In a rasterizer 305,306, the received profile vector data is developed to a bit pattern, and it is returned to the system control section 304. The system control section returns the bit pattern data which 304 received from the rasterizer 305,306 to the application section 307.
[0017] Next, an example is explained to a detail based on the above explanation.
[0018] This example carries out typeface data file division, aims at enabling acquisition of typeface data by installing only a required part, and is briefly explained about the division approach using drawing 4 .
[0019] Drawing 4 expresses the configuration of a division file briefly. In addition, in order to simplify explanation, Typeface A shall constitute one typeface from character codes 0–600.
[0020] This typeface A should be divided into A3 [of 0–100] of A1,101–199 of A2,200–500, and four parts of 501–600. In this typeface A, A3 part is data which generally are not used, and is a part which needs disk memory most.
[0021] The part which should show such divided typeface information at the time of typeface install, and should install it is made to choose by a user's hand (keyboard). As a result of being chosen, only the part needed is installed and the typeface information file 500 which described the items of the typeface of the installed result which is shown in drawing 5 A is created on external storage 109. In addition, a situation is shown as a result of the typeface file installed in drawing 5 B.
[0022] The information which shows the items of the typeface which read information and was installed out of the typeface information file from the typeface information file 500 at the time of a system startup is taken out, and a typeface management information table as shown in drawing 6 is created on the typeface information management field 205.
[0023] The flow of typeface management information table creation is explained using drawing 7 . In addition, this drawing is processing about the typeface under initial processing when a power source is supplied to this system, and **** procedure is stored in external storage 109.
[0024] First, a typeface information file is opened at step S702, at the following step S703, information on a typeface name is acquired from the opened typeface information file, and the information on a typeface name is written in a typeface management information table.
[0025] In step S704, a file name is acquired from a typeface information file, and the information on a file name is written in a typeface management information table. And in step S705, the file open shop operation for considering as an accessible condition is performed based on the file name of the typeface gained previously. In step S706, from the typeface file which became accessible in step S705, the information about the initiation code of the typeface data stored in the file and a termination code is read, and it writes in a typeface management information table.
[0026] When the file of the same typeface investigates whether it still exists to the description in a typeface information file and processing exists in it in step S707, acquisition of return and the following file name and an update process of a table are performed to step S704.
[0027] Moreover, when it is judged that the file of the same typeface does not exist at step S707, it progresses to step S708 and investigates whether an unsettled typeface exists in a typeface management file, and when it exists, it returns to S703 and processing to the following typeface is performed. Moreover, if it judges that the creation of a table to all typefaces was completed, it will progress to step S709, the typeface information file which performed open shop operation in previous step S702 will be closed, and this processing will be finished.
[0028] In the typeface information management field 205, a typeface management information table as shown in drawing 6 will be created by the above processing.
[0029] Next, flow until it gains the typeface data to need using the typeface management information table of drawing 6 is explained according to the flow chart of drawing 8 R> 8.
[0030] Alphabetic data income processing is started in step S801.

[0031] In step S802, the information (a character code, typeface information) about the alphabetic character acquired from the AP section 307 in drawing 3 is acquired. In the following step S803, the head of the typeface management information table of drawing 6 carries out typeface retrieval (in the initial stage, the pointer is pointing to the head typeface in a table), and it confirms whether to be a corresponding typeface. When it exists and does not exist to step S804, it progresses to step S805.
[0032] If it judges that it is not a corresponding typeface and processing progresses to step S805, it judges whether other typefaces exist, and if there are other typefaces, a pointer will be carried forward there.
[0033] Moreover, if a corresponding typeface can be discovered and processing progresses to step S804, and it puts whether the character code demanded on the typeface management information table is within the limits of [which is managed] it in another way, it will judge whether the character code is managed. Since the pointer in this time is put on the head location of drawing 6 , it will be judged whether the demanded character code is within the limits of "0" to "100."
[0034] Now, when it is judged that it does not exist, it progresses to step S806 and judges whether the file of the same typeface as the next line of the attention line in a table is managed. If it is, one pointer will be carried forward and it will return to step S804. Moreover, when it is judged that the next file does not exist, alphabetic-character-less processing is performed.
[0035] When it is judged that the character code of the demanded typeface is managed on the other hand, it progresses to step S807, and a file name is acquired from the attention line in a typeface management information table, and in step S808, typeface data are gained from the file based on the acquired file name, and it progresses to a post process S810.
[0036] While being able to stop the memory consumption of external storage 109 according to this example since only the required character code range of a required typeface is installable in a system as explained above, since management of an unnecessary part is not performed, it also becomes possible to gather processing speed.
[0037] The <explanation of 2nd example> above-mentioned example

---

[Translation done.]

DESCRIPTION OF DRAWINGS

[Brief Description of the Drawings]
[Drawing 1] It is a system configuration Fig. in an example.
[Drawing 2] It is the memory map of RAM in an example.
[Drawing 3] It is drawing showing the example of a configuration of character-manipulation equipment.
[Drawing 4] It is drawing showing the example of a configuration of the division file in the 1st example.
[Drawing 5 A] It is drawing showing an example of the contents of the typeface information file in the 1st example.
[Drawing 5 B] It is drawing showing the example of a configuration of the typeface file after install in the 1st example.
[Drawing 6] It is drawing showing the example of contents of the typeface management information table in the 1st example.
[Drawing 7] It is the flow chart which shows the procedure of the typeface management information table creation processing in the 1st example.
[Drawing 8] It is the flow chart which shows the flow of the alphabetic character data acquisition processing in the 1st example.
[Drawing 9] It is drawing showing the example of an initial state of the typeface management information table in the 2nd example.
[Drawing 10] It is the flow chart which shows the flow of the alphabetic data income processing in the 2nd example.
[Drawing 11] It is drawing showing the in-between example of a condition of the typeface management information table in the 2nd example.
[Drawing 12] It is drawing showing the example of contents of the typeface information file updated in the 2nd example.
[Drawing 13] It is drawing showing the example of a condition of the typeface management information table at the time of starting from the contents of the updated typeface information file in the 2nd example.
[Drawing 14] It is drawing showing the example of a configuration of the division typeface file in the 3rd example.
[Drawing 15] It is drawing showing the example of contents of the typeface information file in the 3rd example.
[Drawing 16] It is drawing showing the example of contents of the typeface management information table in the 3rd example.
[Drawing 17] It is the flow chart which shows the flow of creation processing of the typeface management information table in the 3rd example.
[Drawing 18] It is drawing showing the example of contents of the typeface information file in the 4th example.
[Drawing 19] It is drawing showing the example of contents of the typeface management information table in the 4th example.
[Drawing 20] It is drawing showing the example of a configuration of the division file in the 5th

example.
[Drawing 21] It is drawing showing the example of contents of the typeface file in the 5th example.
[Drawing 22] It is drawing showing the example of contents of the typeface management information table in the 5th example.
[Drawing 23] It is drawing showing ****** of the division typeface file in the 6th example.
[Drawing 24] It is drawing showing the example of contents of the typeface information file in the 6th example.
[Drawing 25] It is drawing showing the example of contents of the typeface management information table in the 6th example.
[Drawing 26] It is drawing showing a format of the style width-of-face weight information in the 7th example.
[Drawing 27] It is the flow chart which shows the typeface installation procedure in the 7th example.
[Drawing 28] It is drawing showing the contents of the install media in the 7th example, and the contents of the install data file in it.
[Drawing 29] It is drawing showing the example of contents of the typeface information file in the 7th example.
[Drawing 30] It is drawing showing the example of contents of the tag file in the 7th example.
[Drawing 31] It is drawing showing the example of contents of the typeface file in the 7th example.
[Drawing 32] It is drawing showing the alphabetic character structure of using for the file name of the typeface physical file in the 8th example.
[Drawing 33] The structure of the file name of the typeface logic data file of the typeface data in the 8th example is shown.
[Drawing 34] It is drawing showing the various character strings showing the identification information of the physical typeface in the 8th example.
[Drawing 35] It is drawing showing the various character strings showing the identification information of the typeface logic data file in the 8th example.
[Drawing 36] It is drawing showing the semantic content of the style width-of-face information in the 8th example.
[Drawing 37] It is drawing showing the semantic content of the wait information in the 8th example.

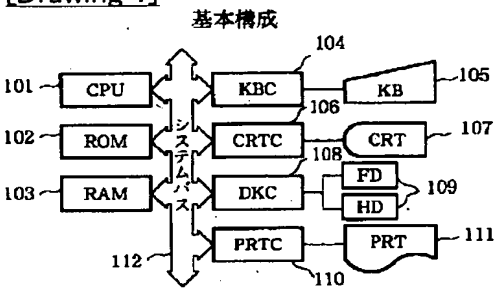
[Drawing 38] It is drawing showing the relation of the size support field in the 8th example.
[Drawing 39] It is the flow chart which shows the contents of processing concerning the reduced-character pattern generation in the 9th example.
[Drawing 40] It is drawing showing the trouble of the contraction processing in the 9th example.
[Drawing 41] It is drawing showing the contraction processing after the amendment in the 9th example.
[Drawing 42] It is drawing showing a format of the name of the typeface information file in the 10th example.
[Drawing 43] It is drawing showing the internal structure of the typeface information file in the 10th example.
[Drawing 44] It is drawing showing the structure of the typeface storing information file in the 10th example.
[Drawing 45] It is the flow chart which shows the procedure of the typeface registration in the 10th example.
[Description of Notations]
101 CPU
102 ROM
103 RAM
104 Keyboard Control Section (KBC)
105 Keyboard (KB)
106 Display Control Section (CRTC)
107 Display
108 DKC

109 External Storage
110 Printer Control Section
111 Printer

[Translation done.]

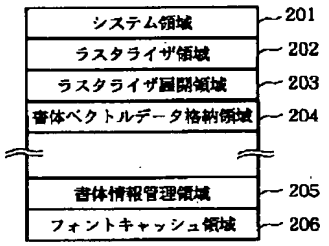
DRAWINGS

[Drawing 1]



[Drawing 2]

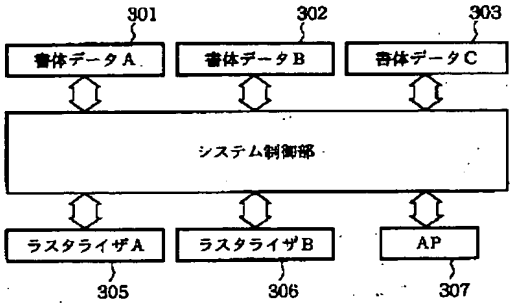
RAM103のメモリマップ



[Drawing 3]

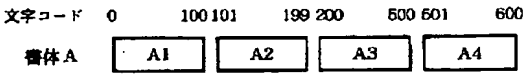
第 3 図

文字データを扱うシステムの構成



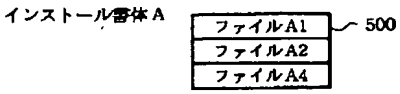
[Drawing 4]

分割書体ファイルの構成



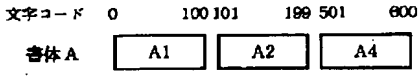
[Drawing 5 A]

書体情報ファイル



[Drawing 5 B]

インストール後の書体ファイルの構成



[Drawing 6]

書体管理情報テーブル

書体名	開始コード	終了コード	ファイル名
A	0	100	A1
A	101	199	A2
A	501	600	A4

[Drawing 9]

書体管理情報テーブル

書体名	開始コード	終了コード	ファイル名	アクセス回数
A	0	100	A1	0
A	101	199	A2	0
A	200	500	A3	0
A	501	600	A4	0

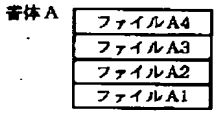
[Drawing 11]

アクセス後の書体管理情報テーブル

書体名	開始コード	終了コード	ファイル名	アクセス回数
A	0	100	A1	1
A	101	199	A2	8
A	200	500	A3	200
A	501	600	A4	1000

[Drawing 12]

並び変えられた書体情報ファイル



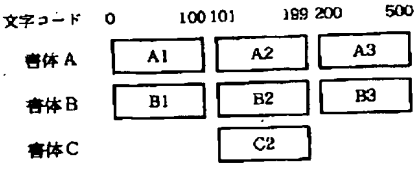
[Drawing 13]

次回起動時の書体管理情報テーブル

書体名	開始コード	終了コード	ファイル名	アクセス回数
A	501	600	A4	0
A	200	500	A3	0
A	101	199	A2	0
A	0	100	A1	0

[Drawing 14]

分割書体ファイルの構成



[Drawing 15]

書体情報ファイル

書体A	
ファイルA1	従属無し
ファイルA2	書体Cに従属
ファイルA3	従属無し
書体B	
ファイルB1	従属無し
ファイルB2	書体Cに従属
ファイルB3	従属無し
書体C（従属書体）	
ファイルC2	従属される

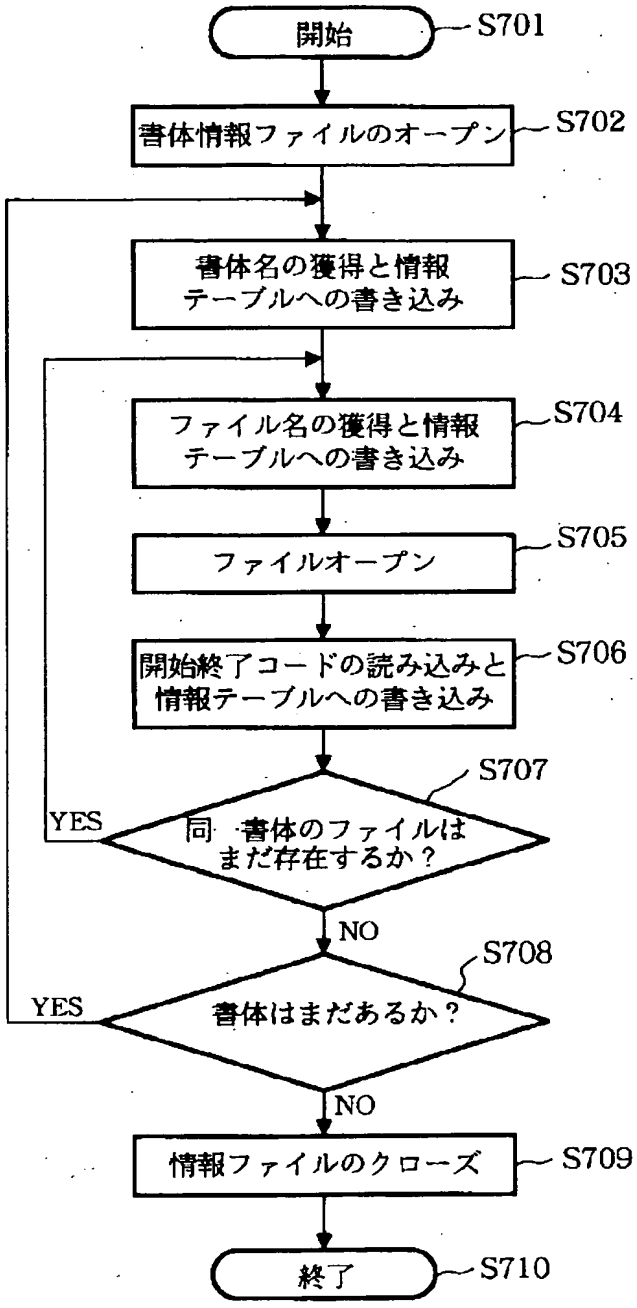
[Drawing 18]

書体情報ファイル

書体A	
ファイルA1	ドライブA
ファイルA2	ドライブB
ファイルA3	ドライブC
ファイルA4	ドライブD

[Drawing 7]

書体管理情報テーブル作成の流れ



[Drawing 16]

書体管理情報テーブル

書体名	開始コード	終了コード	ファイル名
A	0	100	A1
A	101	199	C2
A	200	500	A3
B	0	100	B1
B	101	199	C2
B	200	500	B3

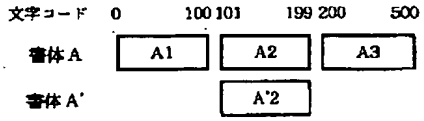
[Drawing 19]

書体管理情報テーブル

書体名	開始コード	終了コード	ファイル名	ドライブ情報
A	0	100	A1	A
A	101	199	A2	B
A	200	500	A3	C
A	501	600	A4	D

[Drawing 20]

分割書体ファイルの構成



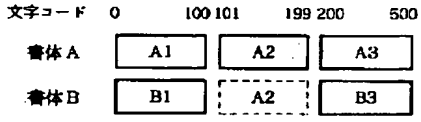
[Drawing 21]

書体情報ファイル

書体A	
キャラクタセット1	ファイルA1
キャラクタセット1	ファイルA2
キャラクタセット1	ファイルA3
書体A'	
キャラクタセット2	ファイルA'2

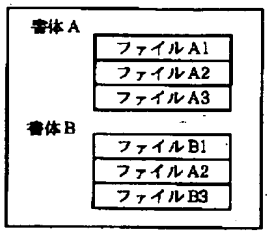
[Drawing 23]

分割書体ファイルの構成

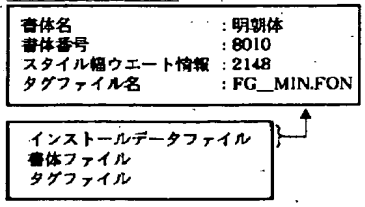


[Drawing 24]

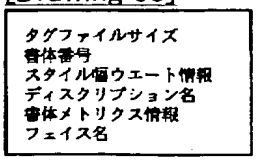
書体情報ファイル



[Drawing 28]



[Drawing 30]



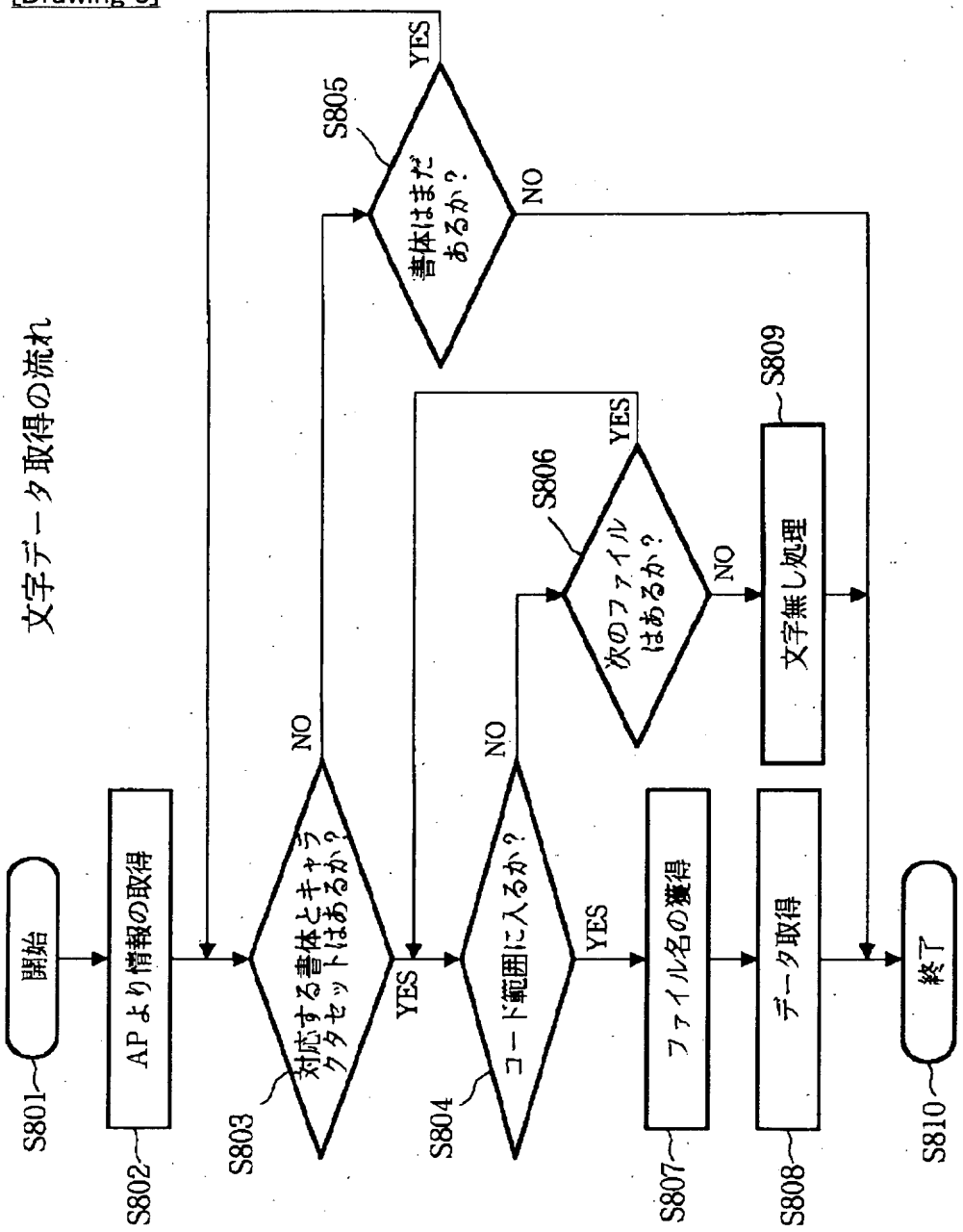
[Drawing 31]

書体構成情報（ファイルサイズ）
書体データ（書体開始コード、文字数）
書体番号
スタイル幅ウエート情報

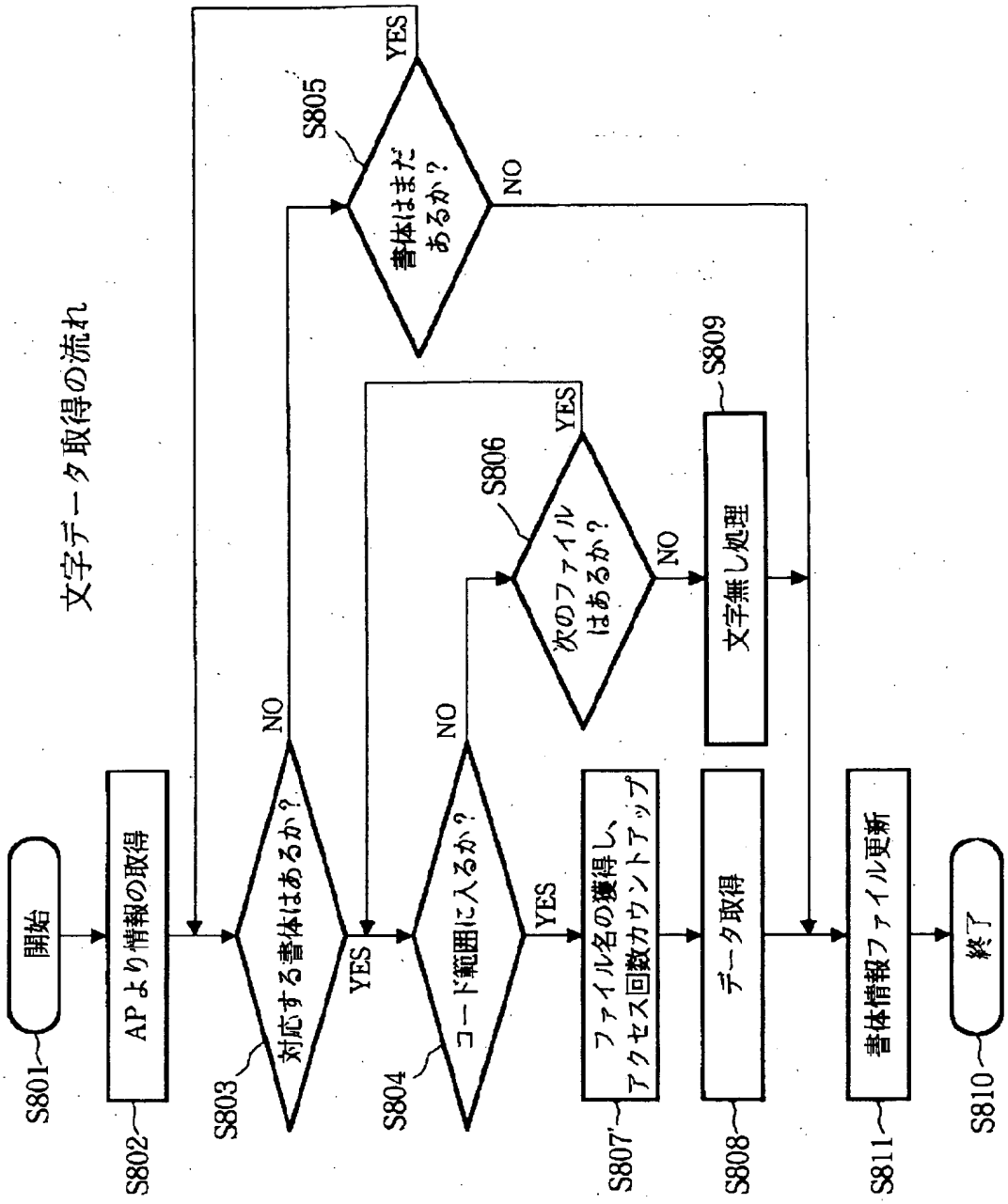
[Drawing 32]

①①②③④④④④.⑤⑤⑤

[Drawing 8]

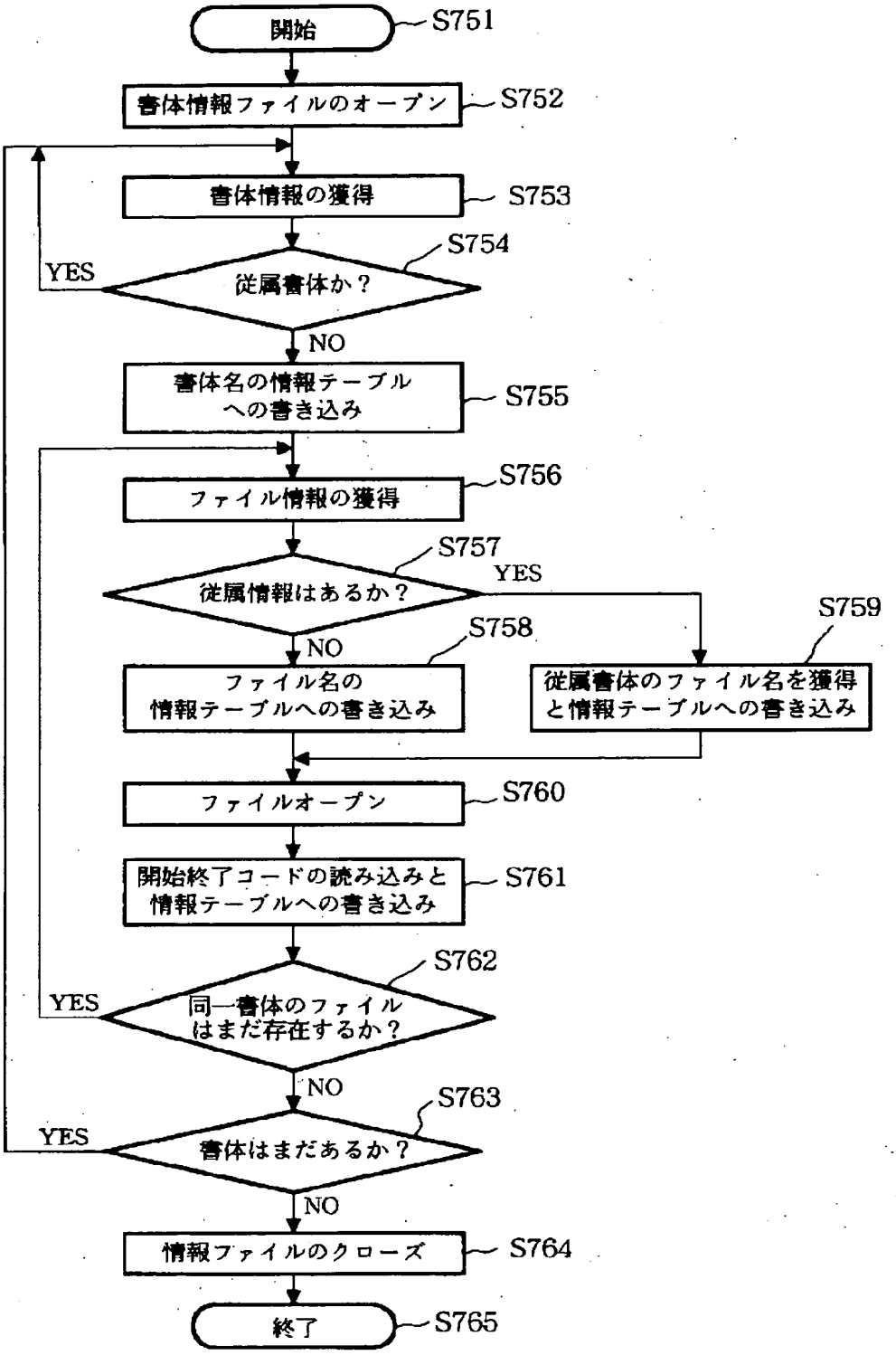


[Drawing 10]


文字データ取得の流れ
開始
S801
APより情報の取得
S802
対応する書体はあるか？
S803
NO
YES
書体はまだあるか？
S805
YES
NO
コード範囲に入るか？
S804
NO
YES
次のファイルはあるか？
S806
YES
NO
文字無し処理
S809
ファイル名の獲得し、アクセス回数カウントアップ
S807
データ取得
S808
書体情報ファイル更新
S811
終了
S810

[Drawing 17]

書体管理情報テーブル作成の流れ



[Drawing 33]

[Drawing 34]

MI	:	明朝体
GO	:	ゴシック体
RG	:	丸ゴシック体
KA	:	楷書体
KY	:	教科書体

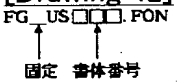
[Drawing 35]

MIN	:	明朝体
GOT	:	ゴシック体
RGO	:	丸ゴシック体
KAI	:	楷書体
KYO	:	教科書体

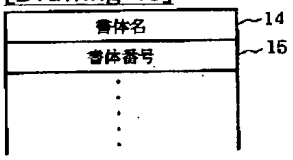
[Drawing 37]

特細	:	1
極細	:	2
細	:	3
中細	:	4
中	:	5
中太	:	6
太	:	7
極太	:	8
特太	:	9

[Drawing 42]



[Drawing 43]



[Drawing 22]

書体管理情報テーブル

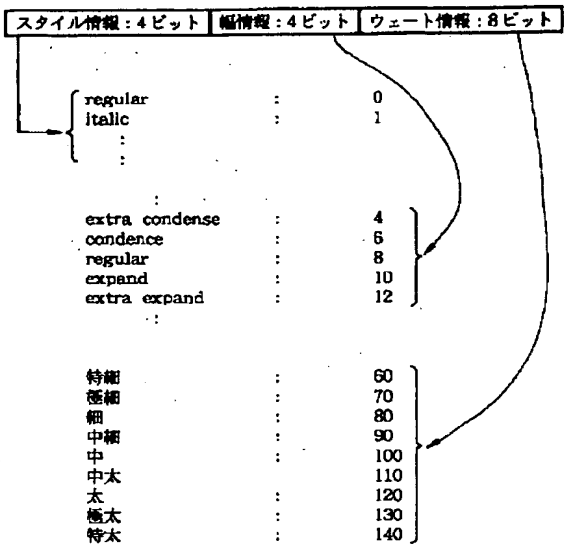
書体名	対応キャラクタセット	開始コード	終了コード	ファイル名
A	1	0	100	A1
A	1	101	199	A2
A	1	200	500	A3
A'	2	101	199	A'2

[Drawing 25]

書体管理情報テーブル

書体名	開始コード	終了コード	ファイル名
A	0	100	A1
A	101	199	A2
A	200	500	A3
B	0	100	B1
B	101	199	A2
B	200	500	B3

[Drawing 26]



[Drawing 29]

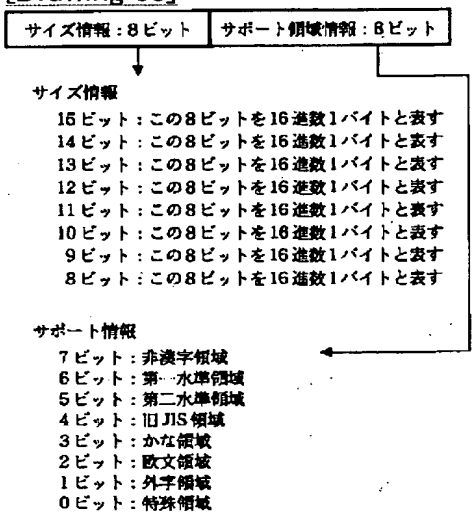
```
NAME＝明朝体,80.10.2148
DEFTAG＝FG__MIN.FON,明朝体
FILE＝C：￥GALLERY￥FONT￥FG__MIN.JFD

NAME＝ゴシック体,81.10.2148
DEFTAG＝FG__GO.FON,ゴシック体
FILE＝C：￥GALLERY￥FONT￥FG__GO.JFD
```

[Drawing 36]

スタイル＼幅	extra condense	condence	regular	expand	extra expand
regular	0	1	2	3	4
italic	5	6	7	8	9

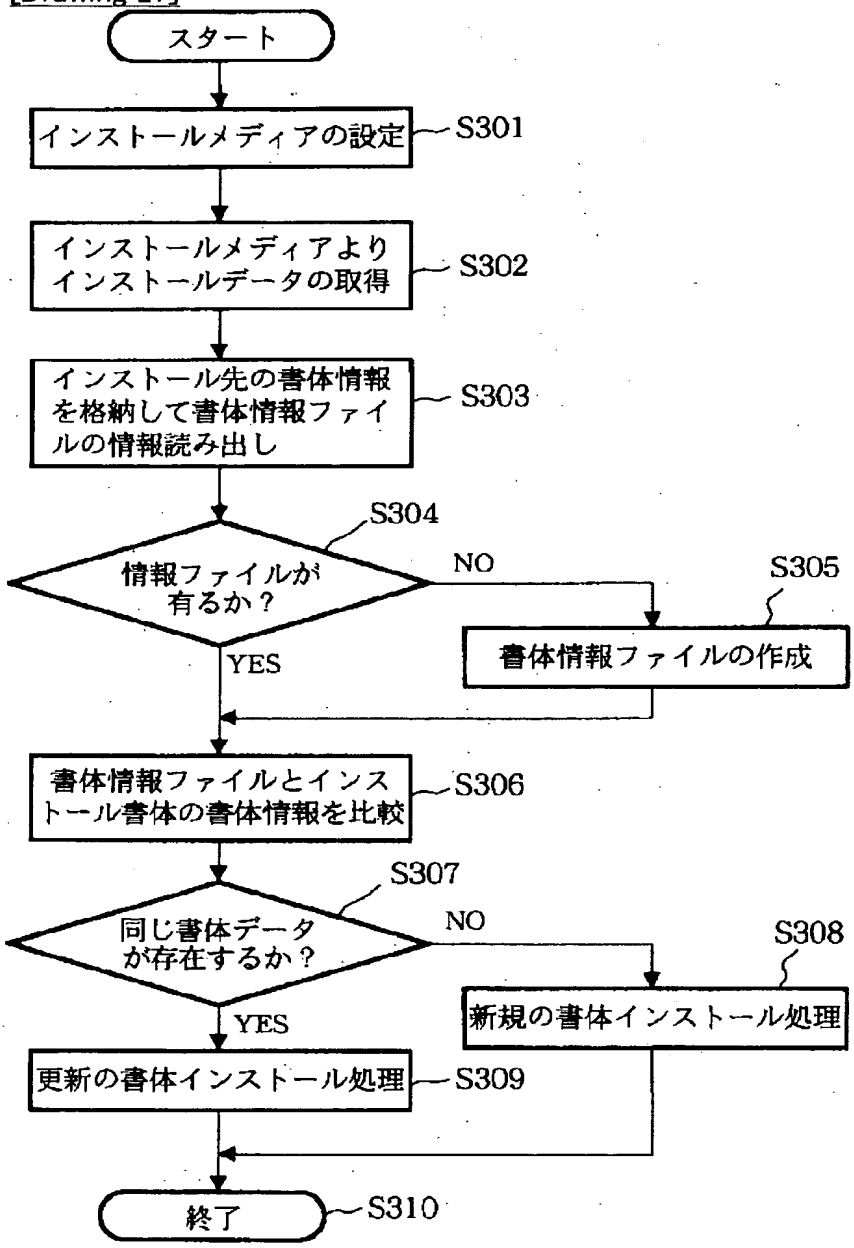
[Drawing 38]



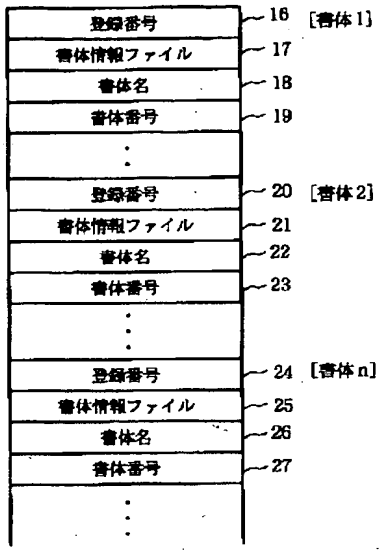
[Drawing 40]

[Drawing 41]

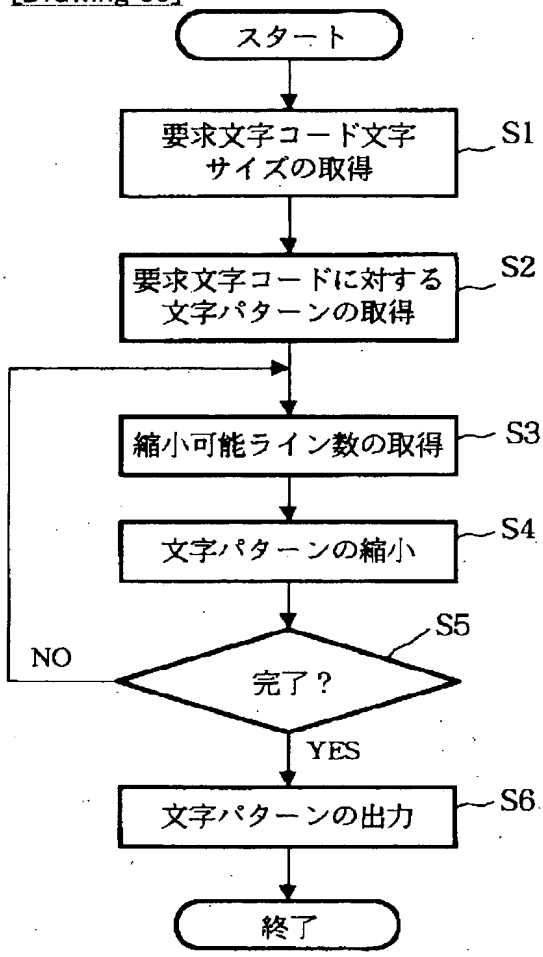
[Drawing 27]



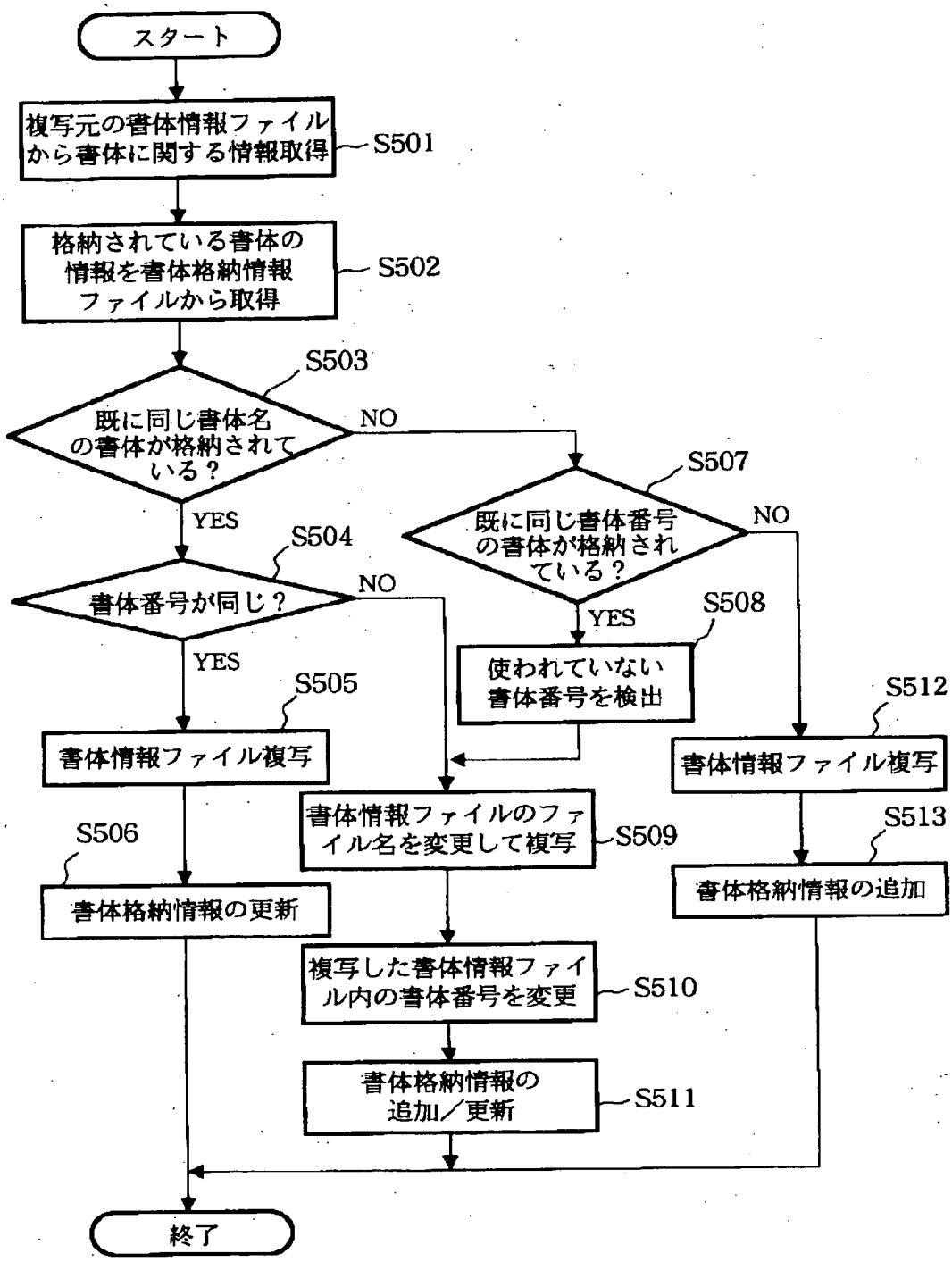
[Drawing 44]



[Drawing 39]



[Drawing 45]



[Translation done.]

(19)日本国特許庁（ＪＰ）　(12)**公開特許公報**（Ａ）　(11)特許出願公開番号

特開平6－266334

(43)公開日　平成６年(1994)９月22日

(51)Int.Cl.[5]	識別記号	庁内整理番号	Ｆ Ｉ	技術表示箇所
Ｇ０９Ｇ　5/22		8121－5G		
Ｇ０６Ｆ　15/72	３５５　Ｕ	9192－5L		

審査請求　未請求　請求項の数10　ＯＬ　（全 23 頁）

(21)出願番号　特願平5－49043

(22)出願日　平成５年(1993)３月10日

(71)出願人　000001007
キヤノン株式会社
東京都大田区下丸子３丁目30番２号

(72)発明者　酒井　哲夫
東京都大田区下丸子３丁目30番２号　キヤノン株式会社内

(72)発明者　佐々木　安彦
東京都大田区下丸子３丁目30番２号　キヤノン株式会社内

(72)発明者　山口　裕成
東京都大田区下丸子３丁目30番２号　キヤノン株式会社内

(74)代理人　弁理士　大塚　康徳　（外１名）

最終頁に続く

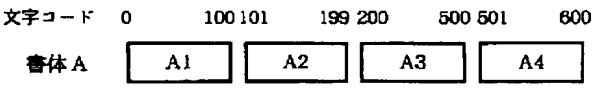
(54)【発明の名称】　書体データ管理方法及び文字パターン発生方法

(57)【要約】

【目的】　柔軟性のある書体データの管理方法を提供する。

【構成】　１つの書体を構成する全文字空間を例えば０～５００とした場合、それをいくつかに分割する。例えば０～１００、１０１から１９９、２００～５００、５０１～６００のコード範囲に分割し、それぞれの空間毎の書体データファイルを管理する。従って、タ鳥羽、コード２００～５００を使用しない場合には、その部分のみを管理外とすることができ、そのファイルを削除することも可能になる。

分割書体ファイルの構成



1

【特許請求の範囲】

【請求項１】　要求された文字に関するデータを書体データファイルから取り出すための書体データ管理方法において、

文字コード空間を複数の部分的なコード空間に分割し、分割された文字コード範囲毎に独立した書体データファイルを管理することを特徴とする書体データ管理方法。

【請求項２】　要求された文字に関するデータを書体データファイルから取り出すための書体データ管理方法において、

文字コード空間を複数の部分的なコード空間に分割し、分割された文字コード範囲毎に独立した書体データファイルを管理し、

各文字コード空間毎の文字データ所得要求の頻度を検出し、

検出した各文字コード空間の頻度順に、個々の書体データファイルの検索順序を変更することを特徴とする書体データ管理方法。

【請求項３】　要求された文字に関するデータを書体データファイルから取り出すための書体データ管理方法において、

文字コード空間を複数の部分的なコード空間に分割し、分割された文字コード範囲毎に独立した書体データファイルを管理し、

注目書体データを構成している部分的書体データファイルの少なくとも１つを他の書体の書体データファイルに従属させることを特徴とする書体データ管理方法。

【請求項４】　要求された文字に関するデータを書体データファイルから取り出すための書体データ管理方法において、

文字コード空間を複数の部分的なコード空間に分割し、分割された文字コード範囲毎に独立したそれぞれの書体データファイルの格納論理的記憶デバイスを記憶保持し、管理することを特徴とする書体データ管理方法。

【請求項５】　要求された文字に関するデータを書体データファイルから取り出すための書体データ管理方法において、

文字コード空間を複数の部分的なコード空間に分割し、分割された文字コード範囲毎に独立した書体データファイルを管理し、

第１の書体と同じで、且つ、同じ文字コード空間における異なるキャラクタセットの書体データファイルを、前記第１の書体として管理することを特徴とする書体データ管理方法。

【請求項６】　書体データファイルから要求された文字に関するデータを取り出す書体データ管理方法において、

文字コード空間を複数の部分的なコード空間に分割し、分割された文字コード範囲毎に独立した書体データファイルを管理する第１の書体を構成する個々の書体データファイルの少なくとも１つに、第２の書体中の対応する文字コード空間の書体データファイルを対応させて管理することを特徴とする書体データ管理方法。

2

【請求項７】　書体データの種別を判別するための書体データ管理方法において、

１つの書体を特定するための、少なくともスタイル、文字幅、書体中における線の太さを表す情報に所定数のビットに割当て、

１つの書体を特定するための情報として、スタイル、文字幅、書体中における線の太さそれぞれを合わせ持つ１つのデータで構築することを特徴とする書体データ管理方法。

【請求項８】　書体データに関するファイルを管理する書体データ管理方法において、

限られたファイルの名称長に、複数の書体の中から１つの書体を特定するための、書体名、スタイル、幅、サイズ、サポートする文字コード範囲を表記文字に符号化して対応させることを特徴とする書体データ管理方法。

【請求項９】　所定のドットサイズの対応する文字パターンから、要求された縮小文字パターンを発生する文字パターン発生方法において、

前記書体のドットサイズの文字パターンに対する前記要求されたサイズによって決定された削除或いは重ね合わせを行う行或いは／及び列の数が所定数以上ある場合、前記所定数の削除或いは重ね合わせとして縮小処理し、該縮小処理によって目的の縮小文字パターンが生成されるまで、前記縮小処理を繰り返すことを特徴とする文字パターン発生方法。

【請求項１０】　文字に関するデータを書体データファイルとして管理する書体データ管理方法において、

１つの書体データファイルを書体名及び書体番号で特定し、

ユーザが作成した書体ファイルにシステムに登録する場合に、

同じ書体名の書体ファイルが既に登録されている場合には、当該登録しよいうとする書体名を変更し、同じ書体番号の書体ファイルがシステムに登録されている場合には未使用の番号を割当てて、

システムが書体を管理する所定のファイルの内容を更新することを特徴とする書体データ管理方法。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は書体データ管理方法及び文字パターン発生方法、例えばパーソナルコンピュータ、ワードプロセッサやデスクトップパブリッシング装置等で使用される各種書体のデータファイルを管理する書体データ管理方法及びそれでもって発生する文字パターンの発生方法に関するものである。

【０００２】

【従来の技術】従来、文字処理システムにおいて管理さ

れる書体データは、管理の容易さなどの理由から、一つのファイル中に書体単位で全文字のデータを格納していた。

【０００３】

【発明が解決しようとする課題】前述の従来方法による書体データ管理方法では、以下に示す問題が有る。

【０００４】あらかじめ書体データの全てがファイル中に格納されており、インストールの対象はあくまでも書体単位での選択しかなかった。このため一般的に使用されることがあまりない書体データまでも同時にインストールされてしまい、ディスクメモリを圧迫する結果となっていた。

【０００５】

【課題を解決するための手段】本発明はかかる従来技術に鑑み成されたものであり、柔軟性のある書体データの管理方法を提供しようとするものである。

【０００６】このため、例えば本発明の１つの書体データ管理方法は以下に示す行程を備える。すなわち、要求された文字に関するデータを書体データファイルから取り出すための書体データ管理方法において、文字コード空間を複数の部分的なコード空間に分割し、分割された文字コード範囲毎に独立した書体データファイルを管理する。

【０００７】

【作用】かかる本発明の構成における構成において、１つの書体データは文字コード空間を複数に分割され、しかも、その分割された部分的なコード空間毎に書体データのファイルを有する。

【０００８】

【実施例】以下、図面を参照し、本願発明について詳細に説明する。

【０００９】まず、パーソナルコンピュータ等の文字データを扱うシステムが、文字の表示或は印字を行う際に、書体データを獲得するまでの動きを簡単に説明する。

【００１０】図１はこれら上記のシステムが動作するための基本的な構成を示すブロック図である。

【００１１】図１において、１０１はＣＰＵ、即ち中央処理装置であり、上記システム全体の起動及び演算処理等を行うものである。１０２は読み出し専用メモリ（ＲＯＭ）であり、システム起動プログラム（ブートプログラム）及び文字パターン・データ等の記憶領域である。１０３はランダムアクセスメモリ（ＲＡＭ）であって、システムプログラム、各種アプリケーションプログラムや、文字パターン発生のためのドライバプログラム等がロードされると共に、各種データの一時的に記憶に用いられる。後述するフローチャートに係るプログラムも勿論、このＲＡＭ１０３にロードされて実行される。１０４はキーボード制御部（ＫＢＣ）であり、１０５のキーボード（ＫＢ）よりキー入力データを受け取り、ＣＰＵ１０１へ伝達する。１０６はディスプレイ装置（ＣＲＴ）１０７の制御を行うＣＲＴ制御部（ＣＲＴＣ）である。１０９はフロッピーディスク装置（ＦＤ）やハードディスク装置（ＨＤ）等の外部記憶装置であって、これらにシステムプログラムやアプリケーションプログラムをはじめとする各種プログラムやフォントデータが格納されている。プログラムの実行時には、これら外部記憶装置１０９に記憶されたプログラムがＲＡＭ１０３にロードされてＣＰＵ１０１により実行されることになる。１１０はプリンタ制御部（ＰＲＴＣ）であり、１１１はＰＲＴＣ１１０の制御の下で、送出されてきたデータに基づく可視画像を記録紙上に形成するプリンタ（ＰＲＴ）である。１１２はシステムバスであり、上述の構成要素間のデータの通路となるべきものである。

【００１２】図２はＲＡＭ１０３のメモリマップを示した図である。２０１はシステムがロードされるシステム領域であり、２０２は文字展開処理部がロードされる領域であり、２０３は文字展開処理部がビットパターンデータを格納する領域である。２０４は文字展開処理時において、１０９の外部記憶装置上に格納されているベクトルパターンデータを参照するために読み込む領域である。２０５は前記システムが動作するための管理情報テーブルを生成するための領域である。２０６は書体ベクトルデータのキャッシュ領域である。

【００１３】図３は前記文字データを扱うシステムの構成を表す図である。

【００１４】図３において、３０１，３０２，３０３は、図１における外部記憶領域１０９に記憶されている様々な書体データを表している。外部記憶装置１０９に記憶される書体データの種類及びその数は実質的に制限はない。

【００１５】３０４はシステム制御部を示し、図２における２０１，２０４，２０５が含まれる。３０５，３０６は文字展開処理部、即ちラスタライザを示し、その文字展開処理機能をソフトウェアのみで実現されていても良く、ハードウェアによって実現されていても良く、ハードウェアとソフトウェアの双方を用いても構わない。また前記システム中に含まれるラスタライザの数は、少なくとも１つ含まれていればその他の制限はない。３０７はシステムに対して文字取得要求を発行するアプリケーション部（例えばワードプロセッサ等のプログラム）を表す。

【００１６】図３において、アプリケーション部３０７は文字取得要求を発する。文字取得要求は、取得する文字を確定するために、文字と１対１に対応した文字コードと書体情報を用いて行う。要求された文字データを取得するためにシステム制御部３０４は書体データ３０１，３０２，３０３から必要なデータを獲得する。獲得したデータがアウトラインフォントを構成する輪郭ベクトルデータであれば、ラスタライザ３０５，３０６にデ

ータを転送する。また、獲得したデータがビットパターンデータであればそのままアプリケーション部３０７へそれを転送する。ラスタライザ３０５，３０６では、受け取った輪郭ベクトルデータをビットパターンへ展開し、それをシステム制御部３０４へ送り返す。システム制御部は３０４がラスタライザ３０５，３０６より受け取ったビットパターンデータをアプリケーション部３０７へ送り返す。

【００１７】次に、以上の説明を踏まえ、実施例について詳細に説明する。

【００１８】本実施例は書体データファイル分割し、必要な部分のみをインストールすることで書体データの獲得を可能とすることを目的としており、その分割方法について、図４を用いて簡単に説明する。

【００１９】図４は分割ファイルの構成を簡単に表したものである。尚、説明を簡単にするため、書体Ａは文字コード０～６００で１つの書体を構成するものとする。

【００２０】この書体Ａを０～１００のＡ１、１０１～１９９のＡ２、２００～５００のＡ３、５０１～６００の４つの部分に分割したものとする。この書体Ａにおいて、Ａ３部分は、一般的に使用されることのないデータであり、かつ最もディスクメモリを必要とする部分である。

【００２１】これらの分割された書体情報を書体インストール時に示し、インストールすべき部分をユーザの手（キーボード）によって選択させる。選択された結果、必要とされる部分のみのインストールを行い、図５Ａに示されるインストールされた結果の書体の内訳を記述した書体情報ファイル５００を外部記憶装置１０９上に作成する。尚、図５Ｂにインストールされた書体ファイルの結果状況を示す。

【００２２】システム立ち上げ時に、書体情報ファイル５００より情報を読み取り、その書体情報ファイル内からインストールされた書体の内訳を示す情報を取り出し、図６に示すような書体管理情報テーブルを書体情報管理領域２０５上に作成する。

【００２３】書体管理情報テーブル作成の流れを図７を用いて説明する。尚、同図は本システムに電源が投入された場合の初期処理中における書体に関する処理であり、そお処理手順は外部記憶装置１０９に格納されている。

【００２４】先ず、ステップＳ７０２で書体情報ファイルをオープンし、次のステップＳ７０３で、そのオープンした書体情報ファイルより書体名の情報の獲得を行い、書体名の情報を書体管理情報テーブルへ書き込む。

【００２５】ステップＳ７０４においては、書体情報ファイルよりファイル名の獲得を行い、ファイル名の情報を書体管理情報テーブルへ書き込む。そして、ステップＳ７０５においては、先に獲得した書体のファイル名を元に、アクセス可能状態とするためのそのファイルオープン処理を行う。ステップＳ７０６においては、ステップＳ７０５においてアクセス可能となった書体ファイルより、そのファイルに格納されている書体データの開始コード、終了コードに関する情報を読み取り、書体管理情報テーブルへ書き込む。

【００２６】処理がステップＳ７０７においては、書体情報ファイル内の記述に同一書体のファイルがまだ存在するかを調べ、存在する場合にはステップＳ７０４に戻り、次のファイル名の獲得とテーブルの更新処理を行う。

【００２７】また、ステップＳ７０７で同一書体のファイルが存在しないと判断した場合には、ステップＳ７０８へ進み、書体管理ファイル内に未処理の書体が存在するかを調べ、存在する場合はＳ７０３に戻って次の書体に対する処理を行う。また、全ての書体に対するテーブルの作成が完了したと判断したら、ステップＳ７０９に進んで、先のステップＳ７０２においてオープン処理を行った書体情報ファイルをクローズし、本処理を終える。

【００２８】以上の処理によって、書体情報管理領域２０５には例えば図６に示すような書体管理情報テーブルが作成されることになる。

【００２９】次に、図６の書体管理情報テーブルを用いて、必要とする書体データを獲得するまでの流れを、図８のフローチャートに従って説明する。

【００３０】ステップＳ８０１において、文字データ所得処理を開始する。

【００３１】ステップＳ８０２においては、図３におけるＡＰ部３０７より取得する文字に関する情報（文字コード，書体情報）を取得する。次のステップＳ８０３においては、図６の書体管理情報テーブルの先頭の書体検索し（初期段階ではポインタはテーブル中の先頭書体を指し示している）、対応する書体であるかチェックを行う。存在する場合はステップＳ８０４へ、存在しない場合はステップＳ８０５へ進む。

【００３２】対応する書体でないと判断し、処理がステップＳ８０５に進むと、他の書体が存在するかどうかを判断し、他の書体があればそこにポインタを進める。

【００３３】また、対応する書体が捜し出せ、処理がステップＳ８０４に進むと、書体管理情報テーブル上に要求された文字コードがその管理されている範囲内にあるかどうか、換言すればその文字コードが管理されているかどうかを判断する。この時点でのポインタは図６の先頭位置に置かれているので、要求された文字コードが“０”から“１００”の範囲内であるかどうかを判断することになる。

【００３４】さて、存在しないと判断された場合には、ステップＳ８０６に進み、テーブル内の注目行の次の行に同じ書体のファイルが管理されているかどうかを判断する。もしあれば、ポインタを１つ進め、ステップＳ８

０４に戻る。また、次のファイルが存在しないと判断した場合には、文字無し処理を行う。

【００３５】一方、要求された書体の文字コードが管理されていると判断した場合には、ステップＳ８０７に進み、書体管理情報テーブル内の注目行からファイル名を獲得し、ステップＳ８０８において、その獲得したファイル名を元に、そのファイルから書体データを獲得し、終了処理Ｓ８１０へ進む。

【００３６】以上説明したように本実施例によれば、必要な書体の必要な文字コード範囲のみをシステムにインストールできるので、外部記憶装置１０９のメモリ消費量を抑えることができると共に、不要な部分の管理は行わないので処理速度を上げることも可能になる。

【００３７】＜第２の実施例の説明＞上記実施例（第１の実施例）では、例えば書体管理情報テーブルが図６に示すようになっている場合、アプリケーション部から書体Ａの文字コード“５０１”の文字所得要求があったときには、図８におけるステップＳ８０４、ステップＳ８０６を３回ループすることになる。文字コード“５０１”から“６００”までに対する文字所得要求の頻度が全体の要求に対してさほど高くない場合には問題がないが、その頻度が他の文字コードに対して多い場合には処理速度の低下は避けられない。

【００３８】そこで本第２の実施例では、文字所得要求の発生頻度を計数し、次回の起動時の書体管理情報テーブル作成に反映させるものである。

【００３９】尚、本第２の実施例におけるシステム構成も図１に示すものとする。これは後述する第３の実施例以降についても同様である。また、本第２の実施例では、書体データを外部記憶装置１０９にインストールする場合に、図４に示した全ての範囲の書体データＡ１～Ａ４を指定した場合を説明する。また、図８のフローチャートに係るプログラムは当然のことながら外部記憶装置１０９に記憶されているものである。この点も後述する各実施例についても同様である。

【００４０】図９は第２の実施例において作成された書体管理テーブルの初期状態を示している。第１の実施例の図６との違いは、同一書体における個々のコード範囲のアクセス回数を記憶保持する欄が設けられた点である。書体管理情報テーブル自身を作成する手順は第１の実施例と同様であるので、ここでの説明は省略し、文字データ所得処理を図１０に従って説明する。

【００４１】図１０のフローチャートと第１の実施例における図８との相違点は、ステップＳ８０７がステップＳ８０７’となり、ステップＳ８１１が追加された点である。

【００４２】図示において、書体管理テーブルはアプリケーション部から文字データ所得要求があって、要求された文字データが管理テーブルで管理されている場合、処理はステップＳ８０４からステップＳ８０７’に進むのは同じである。ステップＳ８０７に処理が進むと、書体管理情報テーブルからファイル名を獲得すると共に、書体管理情報テーブルの該当するアクセス欄の数値を“１”インクメントする。そして、ステップＳ８０８において、その獲得したファイル名を元に、そのファイルから書体データを獲得し、終了処理Ｓ８１１へ進む。

【００４３】この結果、例えば書体管理情報テーブルは図１１に示すようになる。

【００４４】ステップＳ８１１では、書体管理ファイル内に記憶されている順序を、その時点での管理テーブル内のアクセス回数で並べ変える処理を行い、本処理を終える。

【００４５】例えば、書体管理情報テーブルが図１１に示す状態にあった場合、書体Ａに関してはファイルのアクセス回数がＡ４＞Ａ３＞Ａ２＞Ａ１であるので、書体情報ファイルの中身は図１２に示すようにその順序が変更される。

【００４６】装置起動時には、書体情報ファイルに記憶された順序に従ってテーブルを作成するから、次回の起動時の書体管理情報テーブルは図１３に示すようになる。すなわち、書体Ａにおけるファイルａ４が一番最初に位置することになり、文字データ所所得要求に対するアクセス速度の向上が望める。

【００４７】尚、第２の実施例では、文字データ所得要求が発生する度に、書体情報ファイルを更新したが、システムの電源断処理時に行うことで、処理速度をより高速化することが可能になる。

【００４８】また、システム電源断処理において、各書体を構成している部分ファイル（Ａ１～Ａ４）のアクセス回数も適当なファイルとして保存し、次回のシステム起動時にそのファイルからアクセス回数を所得し、それでもってテーブル内の順序を決定するようにしても良い。

【００４９】以上説明した様に本第２の実施例によれば、アクセス回数の多い文字コード範囲ほど優先して書体情報管理テーブルの先頭に配置するので、文字パターン発生にかかる全体のコストパフォーマンスを高めることが可能になる。

【００５０】＜第３の実施例の説明＞次に第３の実施例を説明する。本第３の実施例では、複数の書体のうちユーザが指定したコード範囲を他の書体で置き換えるものである。

【００５１】図１４は本第３の実施例における分割ファイルの構成を示している。先の第１の実施例における図４のそれと異なる点は複数の書体が分割されていることである。尚、図示において、Ａ１～Ａ３、Ｂ１～Ｂ３及びＣ２はそれぞれファイル名を示している。

【００５２】本第３の実施例では、例えば書体Ａにおける文字コード“１０１”～“１９９”の範囲は書体Ｃのそれを従属させ、例えばアプリケーションから書体Ａの

文字コード１０１の文字データの要求が来たら、ファイルＣ２（実は書体Ｃ）からデータを取り出し、処理するものである。

【００５３】これによれば、例えば、アルファベット、数字、平かな、カタカナ、漢字等が１つの“書体名”を使いながら、上記文字の種別毎に所望とする書体のパターンを発生することが可能になる。また、場合によっては、例えば書体Ａ全体がアウトラインフォントで、書体Ｃがドットパターンである場合には、特定の部分を高速に処理することが可能にもなる。

【００５４】さて、本第３の実施例における書体情報ファイルの内容例を図１５に示す。この書体情報ファイルの作成は、ユーザが自由に作成しても良く、システムが自動的に作成しても良い。或いは、上記第１の実施例で示したように、書体データを本システムにインストールする段階で、ユーザに取捨選択させてもよい。

【００５５】図１５に示すように、書体Ａのファイル Ａ２の部分を書体Ｃの該当する部分に従属させるのであれば、書体Ａにおけるファイル Ａ２は外部記憶装置１０９にインストールすることが不要になるので、外部記憶装置１０９のメモリ消費量を抑えることもできる。更に、ユーザが自分自身の書体を登録する場合にも、文字コードのどの範囲をどの書体で使用するかを決定しさえすればよいので、実質的にユーザ書体で消費されるメモリ量はその従属関係を示した書体情報ファイルのみになる。尚、図１５に示すように、各書体はそれが従属しているのか否かを示す情報が付加されており、１つの書体では文字コード範囲毎にそれが従属しているのか、その範囲に従属するものがあればその書体は何かを示す情報が付加されている。

【００５６】さて、図１５に示すような書体情報ファイルがある場合に、本システムが起動すると、図１６に示すような書体管理情報テーブルがＲＡＭ１０３の書体情報管理領域上に作成される。

【００５７】この書体管理情報テーブルの作成に係る処理を図１７のフローチャートに従って説明する。

【００５８】ステップＳ７５１において本処理が開始されると、書体管理情報テーブルを作成するために書体情報管理領域２０５に或程度の領域を確保する。そして、ステップＳ７５２に進んで、外部記憶装置１０９に保存されている書体情報ファイルをオープンし、次のステップＳ７５３でそのオープンした書体情報ファイルから書体情報を獲得する。

【００５９】ステップＳ７５４に処理が進むと、注目している書体が従属書体かどうかを判断する、従属処理である場合には、ステップＳ７５３に戻り、従属書体ではないと判断したら、ステップＳ７５５に進む。

【００６０】ステップＳ７５５に処理が進むということは、１つの書体に対するテーブル作成開始を意味する。従って、ここでは、その書体名を書体情報管理テーブルに書き込み、以下に説明する処理でその書体に対する文字コード範囲毎の従属関係を明確にしたテーブルを構築していく。

【００６１】ステップＳ７５６では、オープンしている書体情報ファイルから次の情報、すなわち、注目している書体中のコード範囲に対する従属情報を取り出す。そして、ステップＳ７５７では、取りだした文字コード範囲は従属されるのか否か、換言すれば、取りだした情報中に従属情報があるかどうかを判断する。

【００６２】従属情報がなければ、ステップＳ７５８に進み、注目している書体に固有のファイル名を、注目しているコード範囲に対応するよう書体情報管理テーブルに書き込む。

【００６３】また、従属情報があれば、ステップＳ７５９に進んで、その従属する書体の対応するファイル名を取り出し、それを書体情報管理テーブルに書き込む。

【００６４】いずれの場合でも、処理はステップＳ７６０に進み、ステップＳ７５８或いは７５９で獲得されたファイル名を基に、そのファイルをオープンしアクセス可能にする。そして、次のステップＳ７６１において、アクセス可能となって書体ファイルより、そのファイルに格納されている書体データの開始コード及び終了コードに関する情報を読み取り、それを書体情報管理テーブルに書き込む。

【００６５】ステップＳ７６２に処理が進むと、１つの書体に対する処理が終了したかどうかを判断し、終了していないと判断したらステップＳ７５６に戻って上記処理を行う。

【００６６】また、１つの書体に対する全コード範囲に対する処理が終了したと判断した場合には、ステップＳ７６３に進み、次に処理すべき書体があるかどうかを判断する。処理すべき書体が残っていたら、処理はステップＳ７５３に戻り、処理すべき書体が無かったらステップＳ７６４に進む。

【００６７】ステップＳ７６４に処理が進んだ場合には、オープンしている書体情報ファイルをクローズし、ステップＳ７６５で本処理を終了する。

【００６８】以上の結果、例えば図１５に示すような書体情報ファイルがあった場合には、図１６に示すような書体情報管理テーブルが作成されることになる。

【００６９】尚、こうして作成された書体情報管理テーブルを使用して書体データを獲得するときの処理は、先に説明した第１の実施例と同様である。すなわち、アプリケーション部から書体Ａの文字コード“１０１”の書体データ獲得要求があったら、そのテーブルを調べ、書体Ａに対する文字コード“１０１”はファイル名Ｃ２であることが検出されるので、そのファイルＣ２から対応する書体データを取り出し、文字パターンを発生する。この書体データがアウトラインデータである場合には、ラスタライザ３０５に一旦渡してドットパターンを発生

させ、それをアプリケーションに渡すことになる。

【００７０】以上説明したように本第３の実施例によれば、或る１つの書体の所望とするコード範囲を別の書体のデータを置き換えることにより、例えば英文や日本語の混じり合った文章を作成する場合等においては、いちいち書体を切り替えなくともそれぞれにユーザの意志が反映された書体での文字入力が行えるようになる。

【００７１】しかも、インストール時に、或いは任意の時間に、他の書体のデータを従属するよう指示されたコード範囲にもともとあった書体データは実質的に不要になり、その部分をインストール対象から外したり、削除することが可能になるので、外部記憶装置１０９のメモリ消費量を抑えることが可能になる。

【００７２】＜第４の実施例の説明＞第４の実施例を以下に説明する。通常、システムに書体データをインストールする場合、１つの書体全体を１つの論理記憶デバイスに記憶することが必須である。本第４の実施例は、１つの所定の記憶先を論理的に異なるデバイスに記憶することを可能にする。これによって、例えば各論理デバイスが１書体全部を記憶するだけの空き容量を有さない、或いは、各論理デバイスに或る程度の空きよう量を設けたいというような要求に応えるものである。

【００７３】尚、ここで言う、論理記憶デバイスとは、物理的装置をも含む概念であり、例えば１つの物理的な記憶装置を論理的に複数の論理的な装置として区切られた場合、或いは同じ物理的装置であってもパスが異なるものをも含む概念である。

【００７４】本第４の実施例における書体情報ファイルの例を図１８に示す。尚、図示では説明を簡単にするために、１つの書体に対してのみ示した。図示において、Ａ１～Ａ４は上記第１～第３の実施例で説明したようにコード範囲に対応する書体Ａのファイル名である。

【００７５】図示について簡単に説明すると、書体Ａに対する或るコード範囲の書体データは、ファイルＡ１として論理ドライブＡに記憶されていることを示している。

【００７６】さて、本第４の実施例では、かかる書体情報テーブルをシステム起動時に取り出し、その中に記述されたデータに基づいて図１９に示すような書体情報管理テーブルを作成する。

【００７７】第４の実施例における書体情報ファイル及びそれに基づいて作成される書体情報管理テーブルのフォーマットは、第１の実施例の図５Ａ及び図６に示したフォーマットに対して記憶先のドライブ名が記載されている点のみが異なる。当業者であれば、先に説明した第１の実施例のかかる処理内容及び図１８に示した書体情報ファイルの構造に間がみれば容易に推察されるであろう。従って、ここではその説明は省略する。

【００７８】ただし、本第４の実施例では、インストールする時点で書体データを構成する個々のコード範囲のファイルの記憶先を指示するものとして説明したが、勿論、一旦全てを同一記憶ドライブに記憶させ、その後、、必要に応じて適当なコード範囲の書体データを別な記憶装置（例えばより高速な外部記憶装置）に移動させてもよい。

【００７９】以上説明したように本第４の実施例によれば、１つの書体データが１つの論理デバイスの同じ位置にある必要はなく、所望とするデバイスに記憶できるので、外部記憶装置を有効に活用することが可能になる。

【００８０】＜第５の実施例の説明＞次に第５の実施例を説明する。例えば、ある特定範囲の文字のデザインを使用する場合であっても、そのデザインの書体の全範囲の書体データを記憶する必要があるため、外部記憶装置等のメモリ消費量は無視できない。本第５の実施例では、かかる課題を一掃する。

【００８１】図２０は分割ファイルの構成を簡単に示している。図示において、書体Ａは、文字コード０～５００で１つの文字書体全体を構成していることを示しており、０～１００のＡ１、１０１～１９９のＡ２、２００～５００のＡ３の３つ部分的なコード範囲として管理している。書体Ａ’は書体Ａと同じもので、対応するキャラクタセットが異なり、キャラクタセットの切り替え時に変更しなければならない文字データ部分のみを持ち、含まれる文字コードは１０１～１９９のＡ２と同一である。つまり、書体として、Ａ’はＡと同じであるが、使用する文字コードは１０１～１９９のみである。

【００８２】尚、キャラクタセットとは、同じコード範囲（空間）において、例えば平かなとカタカナ、英語のアルファベットとロシア語のアルファベットといった具合に文字表現を言う。

【００８３】上記書体管理を行うため、本第５の実施例における書体情報ファイルは図２１に示すように、書体ファイルとキャラクタセットの関連が記述されている。

【００８４】システム起動時におけるＲＡＭ１０３中の書体管理情報領域２０５に構築される書体管理情報テーブルの作成手順は、図７に示される手順で行われるので、ここでの説明は割愛する。この結果、図２２に示される書体管理情報テーブルが構築される。テーブル中、Ａ１、Ａ２、Ａ３及びＡ２’はそれぞれ書体データＡ，Ａ’を構築するデータのファイル名称である。

【００８５】尚、このようにして作成された書体管理情報テーブルを用いて文字データ所得の流れも、図８に示した手順で行われるが、説明すれば以下の通りである。

【００８６】先ず、ステップＳ８０１において、文字データ所得処理を開始する。

【００８７】ステップＳ８０２においては、図３におけるＡＰ部３０７より取得する文字に関する情報（文字コード，書体情報、キャラクタセット）を取得する。次のステップＳ８０３においては、図６の書体管理情報テーブルを検索し、対応する書体及びキャラクタセットが存

在するかチェックを行う。存在する場合はステップＳ８０４へ、存在しない場合はステップＳ８０５へ進む。

【００８８】ステップＳ８０５では、図２２に示される書体管理情報テーブル上に別の書体がまだ存在するかどうかを判断する。存在する場合にはステップＳ８０３に進み、、テーブル中の別の書体に対するチェックを行う。

【００８９】また、要求された書体がテーブル中に存在すると判断し、ステップＳ８０４に進むと、注目書体中の注目キャラクタセットの開始コード及び終了コード内に、要求されたコードが含まれているかどうかを判断する。尚、コード範囲は、存在しないファイルに関しては、開始コード及び終了コードともに“－１”が格納されているものとしているので、そのコードの文字データが管理されているかどうかは容易に判断できる。

【００９０】さて、ステップＳ８０４において、要求された文字コードが注目しているキャラクタセットの範囲外にあると判断された場合には、ステップＳ８０６に進んで、同一書体に未だチェックしていないキャラクタセットがあるかどうかを判断する。もしあればステップＳ８０４に戻り、再び要求コードに対応するキャラクタセットが存在するかどうかを判断する。また、ない場合には、ステップＳ８０９に進んで、文字無し処理を行う。

【００９１】一方、要求された文字の書体とその文字コードを管理しているキャラクタセットが存在すると判断した場合には、ステップＳ８０７に進み、書体管理情報テーブルからファイル名を獲得し、ステップＳ８０６でそのファイル名のファイルから該当する文字データを所得する。そして、ステップＳ８１０に進んで、処理を終了する。

【００９２】以上説明したように本第５の実施例によれば、同じ書体であっても異なるキャラクタセットの文字が指定された場合であっても、使用する文字のみを含むコード範囲の書体データを有するだけで良く、全文字コードに対応するデータは不要になり、外部記憶装置の記憶領域を有効に活用することができる。

【００９３】＜第６の実施例の説明＞上記第５の実施例では、例えば書体Ａと同じ書体Ａ’を使いながらも、異なるキャラクタセットを使用する場合に、その使用する文字コード範囲の書体データのみを外部記憶装置に記憶させることで、メモリの有効利用を図るものであった。

【００９４】本第６の実施例では、異なる書体を記憶し、一方の書体中の特定のコード範囲の書体データとして、他方の書体の同じ範囲の書体データを活用する例である。この結果、共通して使用する部分の書体データを重複して持つ必要を無くし、メモリの効率的使用を可能とするものである。

【００９５】図２３は本第６の実施例における分割ファイルの構成を簡単に示している。図示において、書体Ａは、文字コード０～５００で１つの文字書体全体を構成

していることを示しており、０～１００のＡ１、１０１～１９９のＡ２、２００～５００のＡ３の３つ部分的なコード範囲として管理している。

【００９６】また、書体Ｂも書体Ａと同様、全文字コード空間を３つ分割されて管理されており、このうち、文字コード０～１００と、２００～５００は書体Ｂ特有の書体データを使用し、コード１０１～１９９については書体Ａのものを使用する。

【００９７】上記書体管理を行うため、本第５の実施例における書体情報ファイルは図２４に示すように、書体の種類とその書体を構築する分割書体データファイル名で構成されている。

【００９８】システム起動時におけるＲＡＭ１０３中の書体管理情報領域２０５に構築される書体管理情報テーブルの作成手順は、図７に示される手順で行われるので、ここでの説明は割愛するが、いずれにしても図２５に示される書体管理情報テーブルがＲＡＭ１０３中の書体情報管理領域２０５中に構築される。

【００９９】文字データ所得処理も、図８に示したフローチャートに従えば良い。すなわち、ＡＰ部から文字データ所得要求があった場合には、その要求のなかで指定された書体データに基づいて書体情報管理テーブルのどの書体が選択されたのかを判断し、次にその書体中の指定された文字コードを管理しているファイル名を所得し、その後、そのファイルから対応する文字データを所得する。

【０１００】＜第７の実施例の説明＞これまで、書体データを判別する場合、書体を表す書体名或いは書体番号、又は、書体名及び書体番号、または、必要によるスタイル情報、幅情報、ウエート情報を別々に用いて判別していたため、その判別処理は複雑にならざるを得ず、時間がかかるという問題があった。

【０１０１】そこで、本第７の実施例では、かかる判別を簡単にしかも高速に行うことを可能にする例を説明する。

【０１０２】図２６は、本第７の実施例におけるスタイル幅ウエート情報のフォーマットを示している。図示の如く、スタイル情報として４ビット（値として０～１５）、幅情報として４ビット、ウエート情報として８ビット（０～２５５）が割り当てられている。スタイル情報としては、“０”はregular、“１”はitalic、…として定義した。また、幅情報としては、extra condence, condence, regular, expand, extra expandに４，６，８，１０，１２を対応させている。そして、ウエート情報としては、特細、極細、細、中細、中、中太、太、極太、特太に、６０，７０，８０，９０，１００，１１０，１２０，１３０，１４０を対応させた。

【０１０３】さて、本実施例におけるインストール手順（外部記憶装置１０９のフロッピーディスクからハードディスクへのインストール）を図２７のフローチャート

に従って説明する。

【０１０４】先ず、ステップＳ３０１において、インストールする書体データとインストールに必要なデータ（インストールデータファイル）が格納されているメディアを設定するよう要求し、そのメディアの指定をユーザに行わせる。

【０１０５】次に、ステップＳ３０２に進んで、インストールする書体データが格納されているインストールデータファイルから書体名情報、書体番号、スタイルウエート情報を読出す。

【０１０６】ステップＳ３０３では、インストール先に既にインストールされている書体情報ファイル内の書体データを読み込む。このとき、読み込むべき書体情報ファイルがインストール先に存在しないとステップＳ３０４で判断したら、その書体ファイルの作成を行う（ステップＳ３０５）。

【０１０７】次に、処理はステップＳ３０６に進んで、インストール先の書体情報ファイルの内容とインストールしようとしている書体情報とを比較する。比較処理の優先順位は、第１に書体番号、第２にスタイル幅ウエート情報、第３の書体名とする。以上の項目が全て同じ場合には無条件で同じ書体である。また、第１番目の項目と第２番目の項目の内容が同じである場合、書体の文字データが同じである為、書体の内容が重ならないようにする処理と、書体名を含めてユニークに処理するかをこの処理で確定する（例えば、ユーザに指示するようにする）。

【０１０８】さて、処理がステップＳ３０７に進むと、本インストール処理によりインストールしようとしている書体データは更新か新規追加であるかを判断する。更新処理とは、先に説明したように３つの項目が全て同じ場合である。

【０１０９】更新処理であると判断した場合には、ステップＳ３０９に進んで、既にインストールされている書体データに上書きするため、書体情報ファイルに記述されている同じ書体の存在位置に上書きする。このとき、ユーザによってインストール先が別個に指示されていれば、その指示された位置に書き込む処理を行う。

【０１１０】また、新規書体インストールであると判断した場合には、ステップＳ３０８に進んで、書体名、書体番号、スタイル幅ウエート情報、タグファイル名、書体情報全てをインストールする。

【０１１１】そして最後に本処理を終了し、呼び出し元に戻る。

【０１１２】図２８はインストールメディアの内容と、その中のインストールデータファイルの中身を示している。

【０１１３】インストールデータファイルには、インストールに必要な情報が記述されている。書体ファイルには、書体情報が格納されている。タグファイルには、書体ファイルの情報より書体を作成するためのデータが格納されている。従って、１つの書体データから複数のタグファイルを用いてタグファイル分の書体が存在すると考えられる。

【０１１４】図２９は、本実施例における書体情報ファイルの内容の一例を示している。“NAME”はディスクリプション名、書体情報（主書体番号、副書体番号、スタイル幅ウエート情報）である。“DEFTAG”はタグファイル名、タグファイルに存在する書体名である。また、“FILE”は書体ディスクのインストール先と書体ファイルのファイル名である。

【０１１５】図３０はタグファイルの内容例を示している。このファイルには、タグファイルのサイズ、書体番号、スタイル幅ウエート情報、ディスクリプション名、書体メトリクス情報に、フェイス名を格納する。この中の書体番号、スタイルウェート情報より書体データへの関連ずけを行う。

【０１１６】図３１は、書体ファイルの内容を示している。このファイルには、書体構成情報（ファイルサイズ）、書体データ（書体開始コード、文字数）、書体番号、スタイル幅ウェート情報が格納される。

【０１１７】以上の如く本第７の実施例によれば、インストール後は、書体情報のスタイル情報、幅情報、ウェート情報を合わせたスタイル幅ウェートを書体を表す情報として持つことで、書体名、書体番号等を比較するよりも簡単に高速に書体の判別が可能になる。

【０１１８】＜第８の実施例の説明＞第８の実施例を説明する。本第８の実施例では、フォント関連のファイル名の最適化を行わせる原理を説明する。例として、本実施例におけるシステム（或いはＯＳ）が管理するファイル名は８文字（８バイト）、拡張子が３文字（３バイト）である例を説明する。

【０１１９】図３２は本第８の実施例における書体物理ファイルのファイル名称に用いる文字構造を示している。

【０１２０】図示において、①は図３４に示す書体識別用の２文字が位置し、②には図３６で示されるスタイル幅情報（０～９）を表す１個の数字が位置する。③には図３７に示されるウェイト情報を表す数字が位置し、④には図３８で示されるサイズ・サポート領域情報である。⑤はファイル識別情報であり、実施例では文字列“ＪＦＤ”を用いる。

【０１２１】また、図３３は、書体データの書体論理データファイルのファイル名の構造を示している。⑥は書体管理情報であり、任意の３文字を用いて表す。①は本ファイルのリリースの順番を示しており、図３５により当てはめる。③は図３７に示すウェイト情報である。⑧はメトリクス情報識別情報であある。本項目は本ファイルのリリースの順番を示す。また、プリンタフォントと同一な情報を持つ場合は、“Ｐ”で表す。⑤はファイル

識別情報であり、この場合は“ＦＯＮ”を使用する。

【０１２２】以下、図３４から図３８について更に詳しく説明する。

【０１２３】図３４は物理書体の識別情報である。ＭＩは明朝体、ＧＯはゴシック体、ＲＧは丸ゴシック体、ＫＡは楷書体、ＫＹは教科書体を表す。図３５におけるＭＩＮは明朝体、ＧＯＴはゴシック体、ＲＧＯは丸ゴシック体、ＫＡＩは楷書体、ＫＹＯは教科書体をそれぞれ表している。

【０１２４】図３６はスタイル幅情報を示している。スタイルとしては、regularとitalicの２種類があり、幅情報としてはextra condence，condence，regular，expand,extra expandがある。この組み合わせによって、例えば図示の如く、regularとextra condenceの場合に“１”を割り当てた。同様に、regularとcondenceに“１”、regularとregularに“２”、regularとexpandに“３”、regularとextraexpandに４を割り当てた。また、italicrとextra condenceの場合に“５”を割り当て、italicとcondenceに“６”、italicとregularに“７”、italicとexpandに“８”、italicとextra expandに“９”を割り当てた。

【０１２５】図３７はウェイト情報を示し、実施例では、特細、極細、細、中細、中、中太、太、極太、特太に、数字１～９を割り当てた。

【０１２６】図３８は実施例におけるサイズ・サポート領域情報である。本データの上位８ビットを用いてサイズデータを１６進数で表す（文字でいうと“１”から“Ｆ”）。尚、アウトラインデータ等のスケーラブル書体データである場合には数字“０”を用いる。また、下位８ビットに各領域を割り当て、対応する領域である場合にはそのビットを“１”にする。実施例ではビット７を非漢字領域、ビット６を第１水準領域、ビット５を第２水準領域、ビット４を旧ＪＩＳ領域、ビット３をかな領域、ビット２を応分領域、ビット１を外字領域、そしてビット０を特殊領域とした。

【０１２７】以上によりファイル名称を決定する。以下、書体情報とその書体情報による決定例を示す。

【０１２８】書体情報｛１．書体名：明朝体、２．スタイル名：regular、３．文字幅情報：condence、４．文字ウェイト情報：中、５．文字サイズ：１００ドット、６．サポート領域、第１水準、７．メトリクス情報識別情報：プリンタフォントと同一なメトリクス情報であるため“Ｐ”、８拡張子によるファイル分類：ＪＦＤ（書体物理データファイル）とＦＯＮ（書体論理データファイル）、９：書体管理環境名称：ＦｏｎｔＭａｎａｇｉｎｇ｝により決定する。

【０１２９】この場合、書体物理データファイルの名称を決定するためには、図３２で示される⑤は“ＪＦＤ”になる。①は図３４によって“ＭＩ”になる。また②は図３６により“１”、③は図３７により“５”になる。サイズ１００は１６進数で表すと、６４Ｈであり、図３８に従ってビット６のみを立てるから下位８ビットは４０Ｈ、従って、④は“６４４０”になる。

【０１３０】つまり、ファイル名“ＭＩ１５６４４０．ＪＦＤ”である。

【０１３１】次に、書体論理データファイルの場合には、拡張子⑤は“ＦＯＮ”、⑥はＦｏｎｔＭａｎａｇｅｒを表すことに統一するので、“ＦＯ＿、①は図３５より“ＭＩＮ”、③は図３７により“５”、⑧は“Ｐ”である。

【０１３２】従って、この書体論理データファイルの名称は“ＦＭ＿ＭＩＮ５Ｐ．ＦＯＮ”である。

【０１３３】以上説明したように本第８の実施例によれば、書体ファイル名称にファイルに格納されているデータを示す情報を表示したことにより、書体ファイルに格納されている情報が解り易くすることができる。

【０１３４】＜第９実施例の説明＞第９実施例について説明する。

【０１３５】通常、文字パターン発生には、先に説明したようにビットマップパターンを予め記憶しておき、それを所得するビットマップフォント方式と、アウトラインデータに基づいて文字の輪郭を形成し、その内部を埋めることで文字パターンを得るアウトラインフォント方式がある。

【０１３６】前者は、比較的小さい文字（ドット構成の小さい文字）に対して有効な方式であり、文字が潰れることなく、且つ、生成するにしても基本となる文字パターンに対して間引きすれば済むので、高速に処理することが可能である。後者は、ラスタライザによって輪郭を描画し、その内部を埋める処理が必要なため、速度は落ちるが、大きい文字（最終的なドット構成の大きい文字）の品位を高めることができる。

【０１３７】ところが、比較的小さい文字を発生するのに適したビットマップフォント方式では、比較的高い確率で要求される文字パターンも、低い確率の文字パターンも、基本となる文字パターンから一律な処理でもって行っており、まだ改善の余地が残っている。

【０１３８】本第９の実施例では、ビットマップフォントによる発生方式を更に改善させるものである。

【０１３９】さて、本第９の実施例では、２０×２０のドットのビットマップフォントがリソースされ、それから４×４～２０×２０の文字パターンを発生する例について説明する。

【０１４０】図３９は実施例におけるビットマップフォントによる文字パターン発生処理の手順を示したフローチャートである。

【０１４１】先ず、ステップＳ１で、要求された文字のコードおよび文字のサイズを所得する。そして、ステップＳ２においては、その要求された文字コードの文字パターン（２０×２０ドット）をＲＯＭ１０２或いは外部

記憶装置１０９より取り出し、ＲＡＭ１０３中の所定エリアに展開する。

【０１４２】ステップＳ３では、１回の処理でデフォルメせずに縮小可能な縦方向のライン数の算出する。要求された文字サイズが先に獲得した文字サイズ（２０×２０）の半分以下の場合には、文字パターンの半分まで、それ以外の場合には、（縮小前の文字サイズ＋２）／５により求める。この式は、２０×２０ドット以下の文字パターンに限定した場合、１回の縮小するライン数を４ライン以下にすることにより、デフォルメ（偏り）を防止することを主眼にした式である。

【０１４３】ステップＳ４に処理が進むと、実際の縮小処理を行う。ここでは、ステップＳ３で求められたライン分の縮小処理を行う。このとき、縮小処理を行う位置は、文字パターンの幅／（縮小ライン数＋１）により求められた等分割位置である。ここで縮小ライン数＝２のときは、分母が奇数になるため、正しく等分割できずデフォルメする可能性がある（図４０参照）。従って２回目の縮小時に前記縮小位置の計算より発生した残りの分だけ位置をずらす補正処理を行う。これを示したのが図４１である。また、縮小ライン数＝４の場合も、分母が奇数となるが、縮小前の文字パターンが２０×２０ドット以下の場合は問題にならないため特に処理は行わない。算出された縦方向のラインでは、その右隣のラインと重ね合わせ（論理和）処理する、案源すれば縮小するラインの位置とその右隣のラインの２ラインで１ラインのドットパターンを生成する。

【０１４４】ステップＳ５では、縮小した結果が、要求されたサイズになったか判定し、全縮小ライン数とステップＳ４で縮小したライン数の差分により判定する。判定の結果、要求されたサイズに満たない場合には、ステップＳ３に戻り、縮小処理を繰り返すことになる。また、要求された文字パターンの作成が完了した場合には、ステップＳ６で要求元に発生した文字パターンを出力する。

【０１４５】尚、上記説明で、１回の縮小処理における縮小するライン数を４以下に抑えるという意味は、例えば、２０×２０の基本サイズから１２×１２のパターンを発生する場合（最終的には８ライン（＝２０－１２）が縮小される）、第１回目において、８ライン全部を縮小するのではなく、第１回目で４ライン分を縮小し、その後で更に４ライン分を縮小することを意味する。因に、先に説明したように２ライン分縮小する場合、すなわち、１８ラインの文字パターンを発生する場合には、その縮小ライン数が４以下であるので問題はないことになる。但し、先に説明した補正処理は行う。

【０１４６】また、本第９の実施例では、横方向に縮小する例を説明したが、縦縦方向への縮小も全く同様に処理することができ、更には縦横共に処理することもできるので、上記内容で本発明が限定されるものではない。

【０１４７】また、使用するビットマップは２０×２０に限定されるものではなく、各々のサイズにおいて縮小する際に効率良く処理が行えるならば、特に１つに限定されるものではない。

【０１４８】また、予め２０×２０、１６×１６、１３×１３、１０×１０等の「（縮小前の文字サイズ＋２）／５」の１回の処理で所得し得る最小サイズの複数ビットマップフォントを用意し、ステップＳ３～Ｓ５のループ数を減らし、且つ、品位の向上に役立てても良い。

【０１４９】以上説明したように本第９の実施例によれば、要求された文字パターンのサイズが本文作成等で最も使用される可能性の高いサイズ（４×４～２０×２０）のみビットマップフォントを使用し、縮小を行う位置等の計算を限定されたサイズに最も有効となるように単純化することにより、パフォーマンスの良い文字パターン縮小が行える。

【０１５０】＜第１０の実施例の説明＞本第１０の実施例について説明する。

【０１５１】上記実施例（例えば第８の実施例）では、ファイル名をそのファイルが有すデータを反映させた。従って、書体を提供する側が、過去に供給した名称とは異なる名称を新たに提供する場合には異なるファイル名を付けるように管理すれば良い。但し、システムのユーザが書体を作成できるようにした場合、供給側ではそのユーザが使用しているファイル名までは管理できない。従って、同じ情報を有する書体が作成される場合があり、一旦同じ情報を含む書体を削除した後に作成した書体の複写を行ったり、同じ情報の該当する書体情報を変更した後に複写を行わねばならない。

【０１５２】本第１０の実施例では、かかる問題を解決する。

【０１５３】図４２は、実施例における書体情報ファイルに付けられるファイル名の一例を示している。図示の如く、本第１０の実施例においては、ファイル名の部分に８文字（８バイト）と、拡張子に３文字（３バイト）を使用している。本実施例では、“ＦＧ＿ＵＳ”は固定のものとし、その後に示した□には書体番号と同じ値が付けられる。但し、ファイル名の付け方はこれに限るものではない。

【０１５４】図４３は書体情報ファイルの内部構造を示している。この書体情報ファイル内には図示の符号１４で示される書体名、符号１５で示される書体番号が格納されている。また、この他には例えば、作成日時やバージョン等の情報も格納されている。

【０１５５】図４４は外部記憶装置１０９に記憶されている書体の格納先を管理する書体格納情報ファイルの構造を示しており、記憶されている全書体についての情報が入っている。この書体格納情報ファイルには、符号１６で示される書体１の登録番号、符号１７の書体１の書体情報ファイル名、１８の書体１の書体名、１９の書体

１の書体番号である。同様に書体２、… 書体ｎそれぞれに対しても同様な情報が格納されている。

【０１５６】図４５のフローチャートに基づいて本第１０の実施例における書体管理手順を説明する。

【０１５７】先ず、ステップＳ５０１において、複写元の書体情報ファイル（ユーザが作成した書体データを管理している書体情報ファイル、４３参照）から書体情報を所得する。次にステップＳ５０２に進んで、システムの管理下におかれている書体の格納情報ファイル（図４４参照）から格納されている書体に関する書体情報を所得する。

【０１５８】ステップＳ５０３においては、先のステップＳ５０１、Ｓ５０２で所得した情報から、同じ書体名の書体が格納されているかどうかを判断する。同じ書体名の書体が既に書体格納情報ファイル内に格納されていると判断した場合にはステップＳ５０４に、その書体名の書体がシステムに登録されていないと判断した場合にはステップＳ５０７に進む。

【０１５９】ここで同じ書体名がシステムに登録されていないと判断され、ステップＳ５０７に進むと、ユーザが作成した書体の書体番号と同じ書体番号がシステムに登録されているかどうかを判断する。ここでも、同じ書体がないと判断したら、処理はステップＳ５１２、Ｓ５１３に進み、システムに登録されている書体情報を記憶している書体格納情報ファイルに、ユーザが作成した書体情報ファイルの内容を複写（追加）し、その書体格納情報ファイルを更新する。

【０１６０】一方、ステップＳ５０３で、同じ書体名の書体が既に登録されていると判断した場合には、ステップＳ５０４に進んで、それれが同じ書体番号であるかどうかを判断する。

【０１６１】同じ書体番号である、すなわち、書体名及び書体番号の両方とも既にシステムに登録されていると判断した場合には、ステップＳ５０５、Ｓ５０６に進んで、書体格納情報ファイル内の該当する位置にユーザが作成した書体情報ファイルの内容で上書きし、書体格納情報ファイルを更新する。

【０１６２】ステップＳ５０７で“ＹＥＳ”であると判断した場合、すなわち、同じ書体名ではないが、同じ書体番号であると判断した場合には、ステップＳ５０８に進んで、新たに登録しようとしている書体の使用されていない書体番号を検出し、ステップＳ５０９に進む。

【０１６３】ステップＳ５０４の判断が“ＮＯ”、つまり、同じ書体名が存在するが、書体番号が異なると判断した場合にもステップＳ５０９に処理が進む。この場合には、書体番号が重複することはないことになる。

【０１６４】いずれにせよ、ステップＳ５０９に処理がくるということは、書体名が同じで、書体番号が異なる場合である。ユーザ側から見て書体名が重要で、同じ書体名が複数あると混乱の素になる。そこで、ステップＳ５０９、Ｓ５１０、Ｓ５１１では、登録しようとしている書体名のファイルを変更（例えば書体名に“USR”を付加する等）し、その変更された書体名を書体格納情報ファイルに追加し、且つ、その書体番号で登録することで、書体格納情報ファイルを更新する。

【０１６５】上記例では、書体名をキー情報として書体の登録状態を判別し書体番号を書体情報ファイルのファイル名と対応させ、書体番号を変更して複写を行うことより、同じ書体情報が含まれる書体を管理することが可能になる。また、本発明はユーザに書体を作成できるようなツールを提供し、前記ツールによる作成された書体を使用するとき等に多大な効果を発揮する。

【０１６６】尚、上記実施例では、書体名をキーにして書体情報を検索し、書体番号を変更するようにしたが、書体番号をキーにして検索し、書体名を変更するようにしても良く、検索に用いるキー情報及び変更する書体情報はこの限りではない。

【０１６７】以上説明した様に本第１０の実施例によれば、書体情報と書体情報ファイルのファイル名を対応させ、同じ書体情報が含まれる書体が検出されたときには、書体情報を変更して複写（登録）を行うようにしたことにより、同じ書体情報が含まれた場合であっても書体を管理することが可能になる。

【０１６８】また、ユーザが書体を作成できるような場合に、同じ情報を有する書体が作成された場合でも、一旦同じ情報を含む書体を削除した後に作成した書体を複写を行ったり、同じ情報に該当する書体情報を変更した後に複写を行ったりする作業の軽減を図ることが可能になる。

【０１６９】上記第１～第１０の実施例に限らず、本発明は、複数の機器から構成されるシステムに適用しても、１つの機器から成る装置に適用しても良い。また、本発明はシステム或は装置にプログラムを供給することによって達成される場合にも適用できることは言うまでもない。

【０１７０】

【発明の効果】以上説明したように本発明によれば、柔軟性のある書体データの管理方法を提供することが可能になる。

【０１７１】

【図面の簡単な説明】

【図１】実施例におけるシステム構成図である。

【図２】実施例におけるＲＡＭのメモリマップである。

【図３】文字処理装置の構成例を示す図である。

【図４】第１の実施例における分割ファイルの構成例を表す図である。

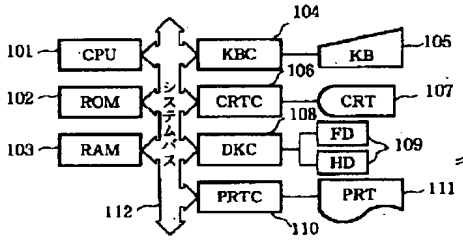
【図５Ａ】第１の実施例における書体情報ファイルの内容の一例を示す図である。

【図５Ｂ】第１の実施例におけるインストール後の書体ファイルの構成例を示す図である。

【図６】第１の実施例における書体管理情報テーブルの内容例を示す図である。
【図７】第１の実施例における書体管理情報テーブル作成処理の手順を示すフローチャートである。
【図８】第１の実施例における文字データ取得処理の流れを示すフローチャートである。
【図９】第２の実施例における書体管理情報テーブルの初期状態例を示す図である。
【図１０】第２の実施例における文字データ所得処理の流れを示すフローチャートである。
【図１１】第２の実施例における書体管理情報テーブルの中間的な状態例を示す図である。
【図１２】第２の実施例で更新された書体情報ファイルの内容例を示す図である。
【図１３】第２の実施例における更新された書体情報ファイルの内容で起動した場合の書体管理情報テーブルの状態例を示す図である。
【図１４】第３の実施例における分割書体ファイルの構成例を示す図である。
【図１５】第３の実施例における書体情報ファイルの内容例を示す図である。
【図１６】第３の実施例における書体管理情報テーブルの内容例を示す図である。
【図１７】第３の実施例における書体管理情報テーブルの作成処理の流れを示すフローチャートである。
【図１８】第４の実施例における書体情報ファイルの内容例を示す図である。
【図１９】第４の実施例における書体管理情報テーブルの内容例を示す図である。
【図２０】第５の実施例における分割ファイルの構成例を示す図である。
【図２１】第５の実施例における書体ファイルの内容例を示す図である。
【図２２】第５の実施例における書体管理情報テーブルの内容例を示す図である。
【図２３】第６の実施例における分割書体ファイルの構例成を示す図である。
【図２４】第６の実施例における書体情報ファイルの内容例を示す図である。
【図２５】第６の実施例における書体管理情報テーブルの内容例を示す図である。
【図２６】第７の実施例におけるスタイル幅ウエート情報のフォーマットを示す図である。
【図２７】第７の実施例における書体インストール手順を示すフローチャートである。
【図２８】第７の実施例におけるインストールメディアの内容と、その中のインストールデータファイルの中身を示す図である。
【図２９】第７の実施例における書体情報ファイルの内容例を示す図である。
【図３０】第７の実施例におけるタグファイルの内容例を示す図である。
【図３１】第７の実施例における書体ファイルの内容例を示す図である。
【図３２】第８の実施例における書体物理ファイルのファイル名称に用いる文字構造を示す図である。
【図３３】第８の実施例における書体データの書体論理データファイルのファイル名の構造を示している
【図３４】第８の実施例における物理書体の識別情報を表す各種文字列を示す図である。
【図３５】第８の実施例における書体論理データファイルの識別情報を表す各種文字列を示す図である。
【図３６】第８の実施例におけるスタイル幅情報の意味内容を示す図である。
【図３７】第８の実施例におけるウェイト情報の意味内容を示す図である。
【図３８】第８の実施例におけるサイズ・サポート領域の関係を示す図である。
【図３９】第９の実施例における縮小文字パターン生成にかかる処理内容を示すフローチャートである。
【図４０】第９の実施例における縮小処理の問題点を示す図である。
【図４１】第９の実施例における補正後の縮小処理を示す図である。
【図４２】第１０の実施例における書体情報ファイルの名称のフォーマットを示す図である。
【図４３】第１０の実施例における書体情報ファイルの内部構造を示す図である。
【図４４】第１０の実施例における書体格納情報ファイルの構造を示す図である。
【図４５】第１０の実施例における書体登録の処理手順を示すフローチャートである。
【符号の説明】
１０１　ＣＰＵ
１０２　ＲＯＭ
１０３　ＲＡＭ
１０４　キーボード制御部（ＫＢＣ）
１０５　キーボード（ＫＢ）
１０６　表示装置制御部（ＣＲＴＣ）
１０７　表示装置
１０８　ＤＫＣ
１０９　外部記憶装置
１１０　プリンタ制御部
１１１　プリンタ

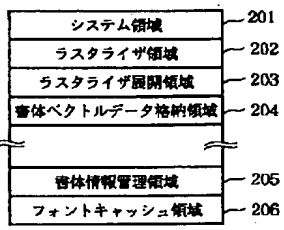
【図１】

基本構成



【図２】

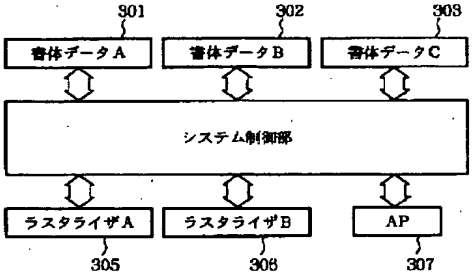
RAM103のメモリマップ



【図３】

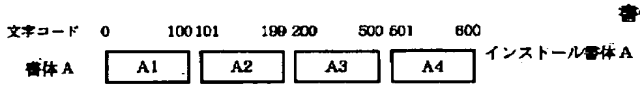
第３図

文字データを扱うシステムの構成



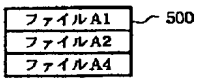
【図４】

分割書体ファイルの構成



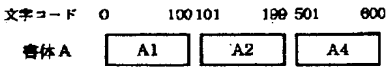
【図５Ａ】

書体情報ファイル



【図５Ｂ】

インストール後の書体ファイルの構成



【図６】

書体管理情報テーブル

書体名	開始コード	終了コード	ファイル名
A	0	100	A1
A	101	199	A2
A	501	600	A4

【図９】

書体管理情報テーブル

書体名	開始コード	終了コード	ファイル名	アクセス回数
A	0	100	A1	0
A	101	199	A2	0
A	200	500	A3	0
A	501	600	A4	0

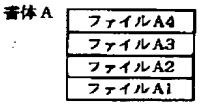
【図１１】

アクセス後の書体管理情報テーブル

書体名	開始コード	終了コード	ファイル名	アクセス回数
A	0	100	A1	1
A	101	199	A2	8
A	200	500	A3	200
A	501	600	A4	1000

【図１２】

並び変えられた書体情報ファイル



【図１５】

書体情報ファイル

書体A	
ファイルA1	従属無し
ファイルA2	書体Cに従属
ファイルA3	従属無し
書体B	
ファイルB1	従属無し
ファイルB2	書体Cに従属
ファイルB3	従属無し
書体C（従属書体）	
ファイルC2	従属される

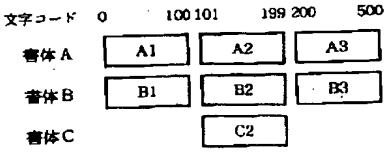
【図１３】

次回起動時の書体管理情報テーブル

書体名	開始コード	終了コード	ファイル名	アクセス回数
A	501	600	A4	0
A	200	500	A3	0
A	101	199	A2	0
A	0	100	A1	0

【図１４】

分割書体ファイルの構成



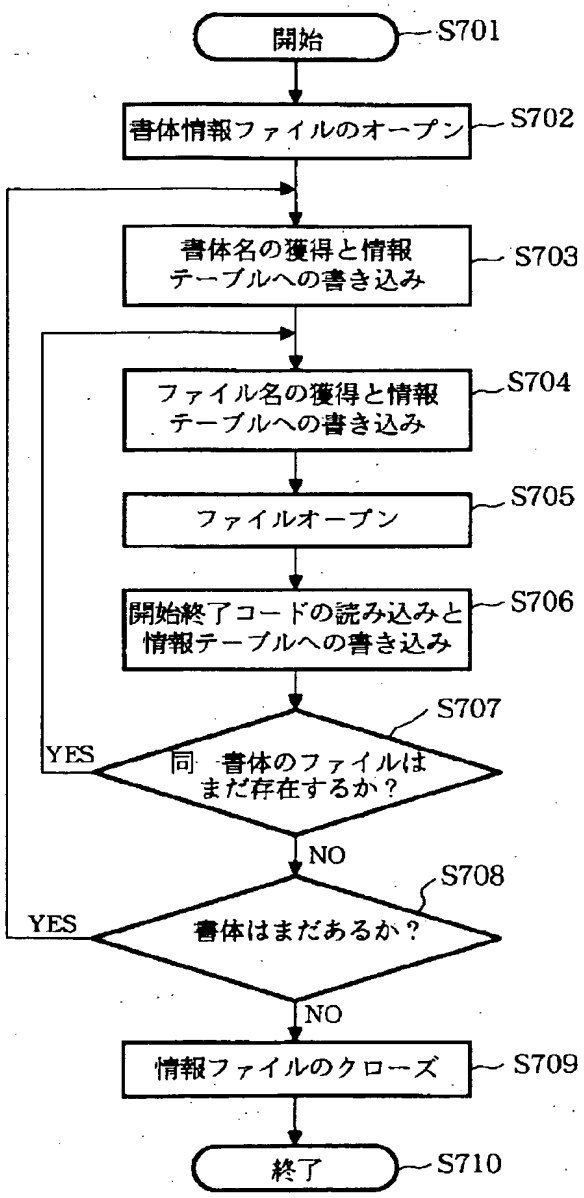
【図１８】

書体情報ファイル

書体A	
ファイルA1	ドライブA
ファイルA2	ドライブB
ファイルA3	ドライブC
ファイルA4	ドライブD

【図７】

書体管理情報テーブル作成の流れ



【図１９】

書体管理情報テーブル

書体名	開始コード	終了コード	ファイル名	ドライブ情報
A	0	100	A1	A
A	101	199	A2	B
A	200	500	A3	C
A	501	600	A4	D

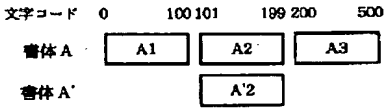
【図１６】

書体管理情報テーブル

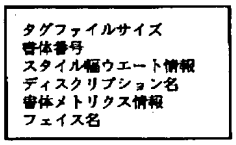
書体名	開始コード	終了コード	ファイル名
A	0	100	A1
A	101	199	C2
A	200	500	A3
B	0	100	B1
B	101	199	C2
B	200	500	B3

【図２０】

分割書体ファイルの構成



【図３０】

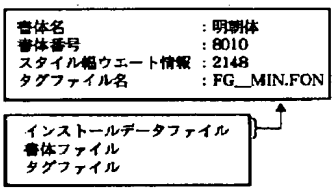


【図２１】

書体情報ファイル

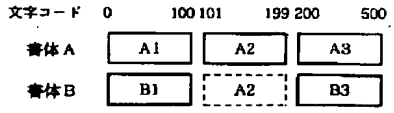
書体A	
キャラクタセット1	ファイルA1
キャラクタセット1	ファイルA2
キャラクタセット1	ファイルA3
書体A'	
キャラクタセット2	ファイルA'2

【図２８】



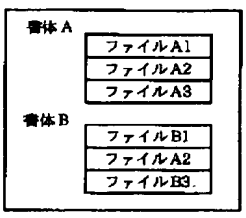
【図２３】

分割書体ファイルの構成

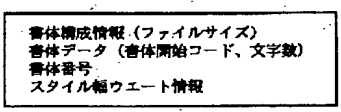


【図２４】

書体情報ファイル

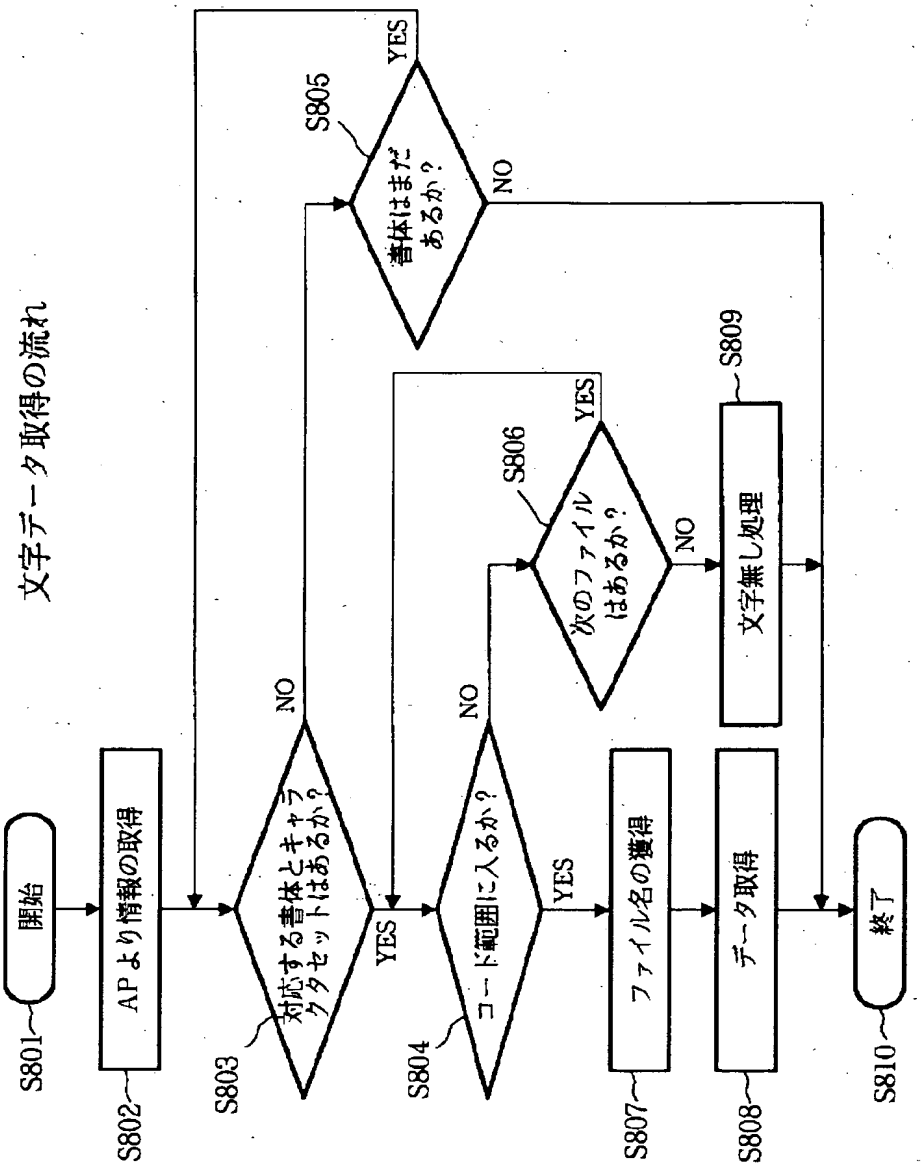


【図３１】

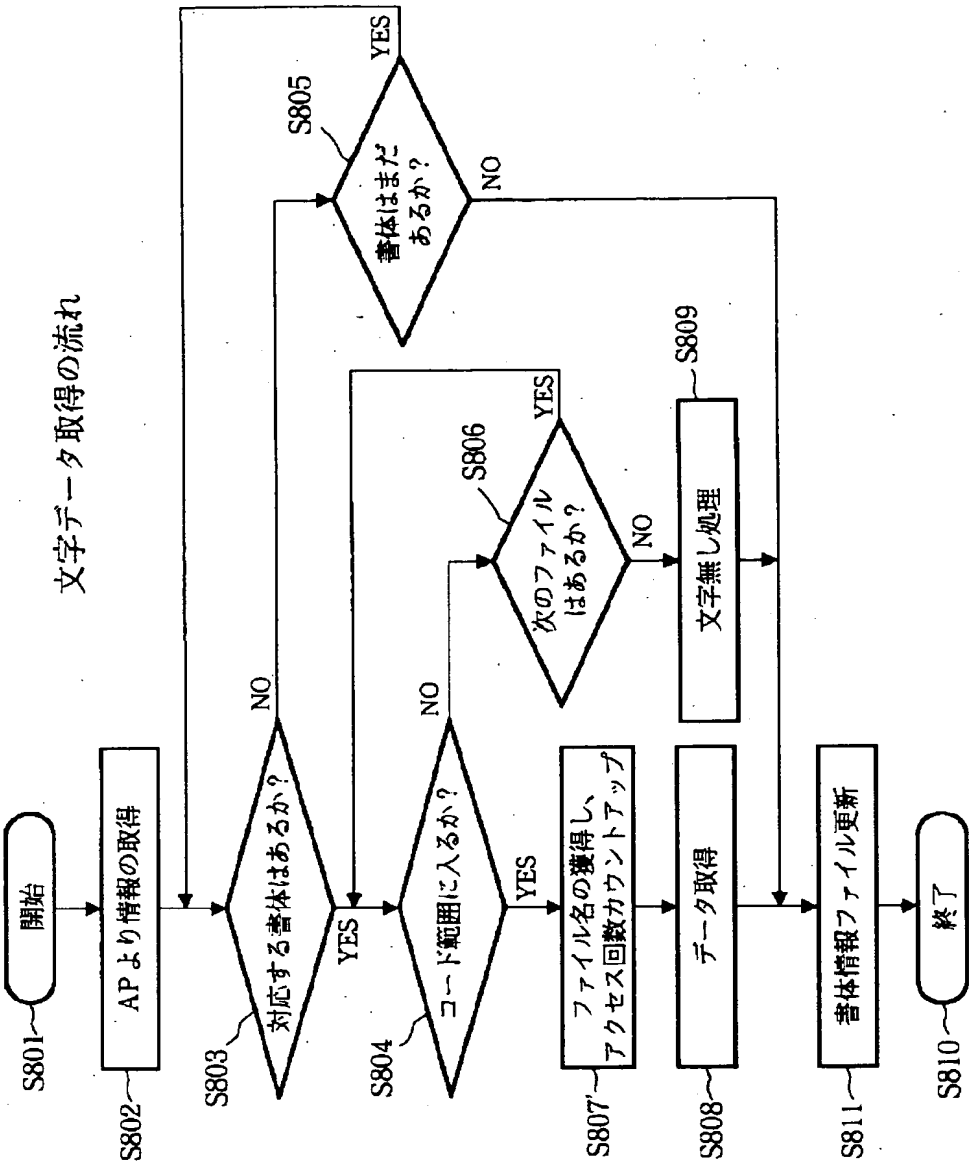


【図３２】

【図８】

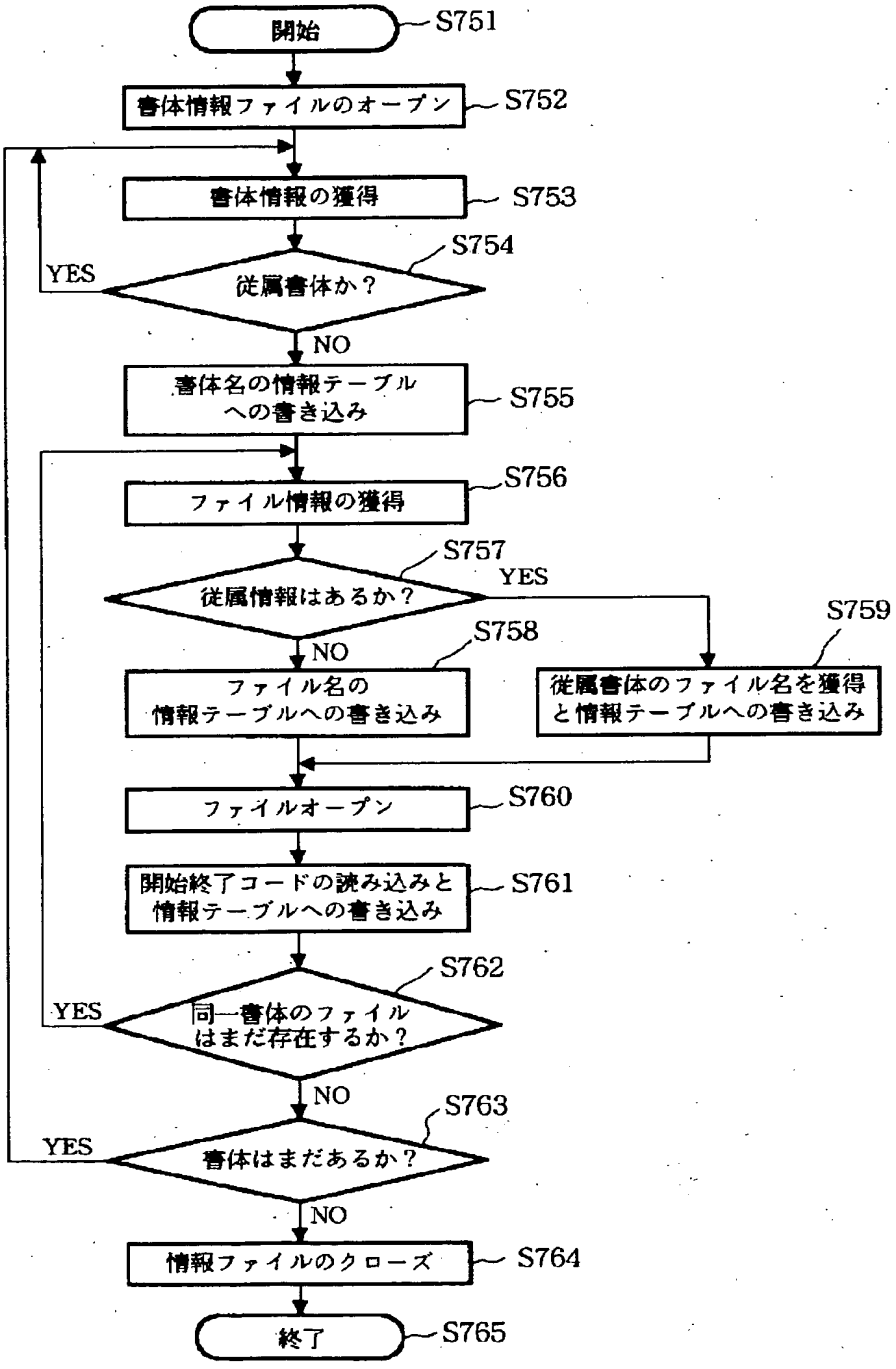

文字データ取得の流れ
開始
S801
APより情報の取得
S802
対応する書体とキャラクタセットはあるか？
S803
NO
YES
書体はまだあるか？
S805
YES
NO
コード範囲に入るか？
S804
NO
YES
次のファイルはあるか？
S806
YES
NO
ファイル名の獲得
S807
データ取得
S808
文字無し処理
S809
終了
S810

【図１０】


文字データ取得の流れ
開始
S801
APより情報の取得
S802
対応する書体はあるか？
S803
NO
YES
コード範囲に入るか？
S804
NO
YES
書体はまだあるか？
S805
YES
NO
次のファイルはあるか？
S806
YES
NO
文字無し処理
S809
ファイル名の獲得し、アクセス回数カウントアップ
S807
データ取得
S808
書体情報ファイル更新
S811
終了
S810

【図１７】

書体管理情報テーブル作成の流れ



【図３３】

⑥ ⑤ ⑥ ① ① ① ④ ⑧ . ⑤ ⑤ ⑤

【図３４】

MI	:	明朝体
GO	:	ゴシック体
RG	:	丸ゴシック体
KA	:	楷書体
KY	:	教科書体

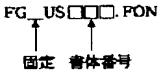
【図３５】

MIN	:	明朝体
GOT	:	ゴシック体
RGO	:	丸ゴシック体
KAI	:	楷書体
KYO	:	教科書体

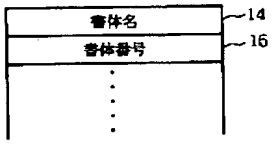
【図３７】

特細	:	1
極細	:	2
細	:	3
中細	:	4
中	:	5
中太	:	6
太	:	7
極太	:	8
特太	:	9

【図４２】



【図４３】



【図２２】

書体管理情報テーブル

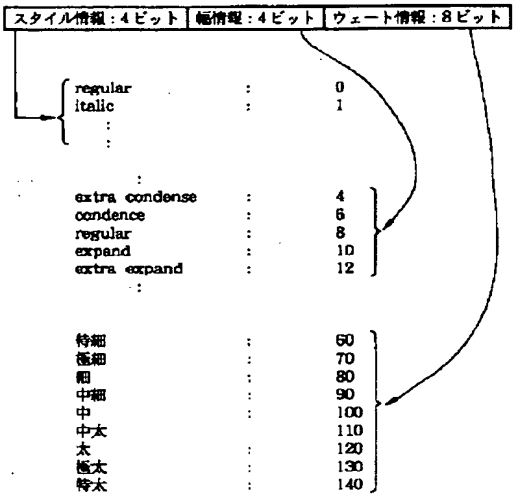
書体名	対応キャラクタセット	開始コード	終了コード	ファイル名
A	1	0	100	A1
A	1	101	199	A2
A	1	200	500	A3
A'	2	101	199	A'2

【図２５】

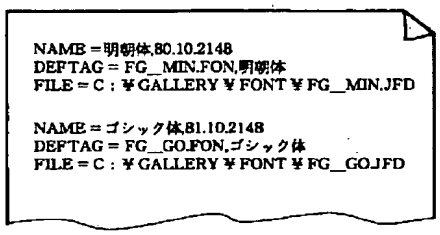
書体管理情報テーブル

書体名	開始コード	終了コード	ファイル名
A	0	100	A1
A	101	199	A2
A	200	500	A3
B	0	100	B1
B	101	199	A2
B	200	500	B3

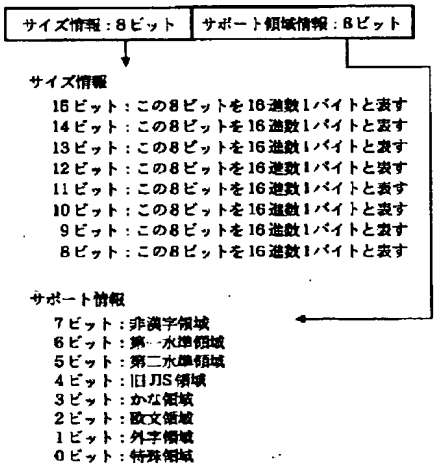
【図２６】



【図２９】



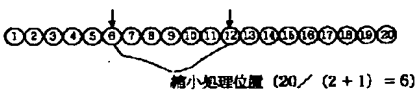
【図３８】



【図3-6】

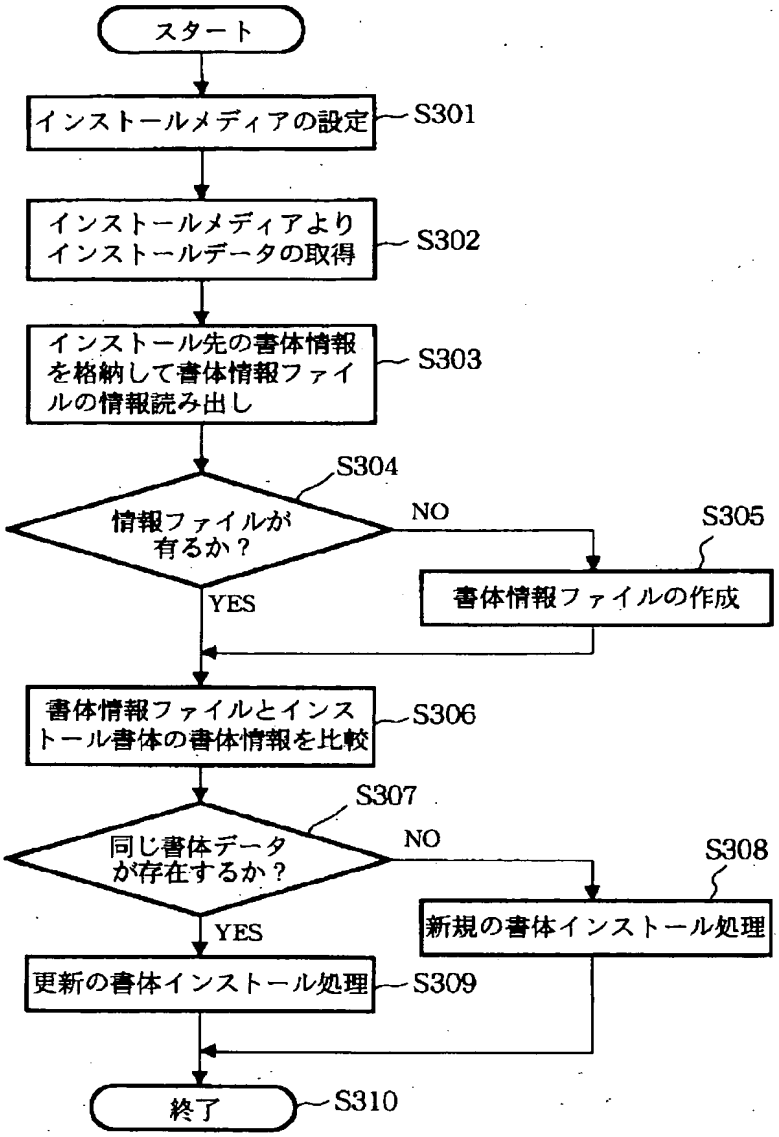
スタイル＼幅	extra condense	condence	regular	expand	extra expand
regular	0	1	2	3	4
italic	5	6	7	8	9

【図４０】

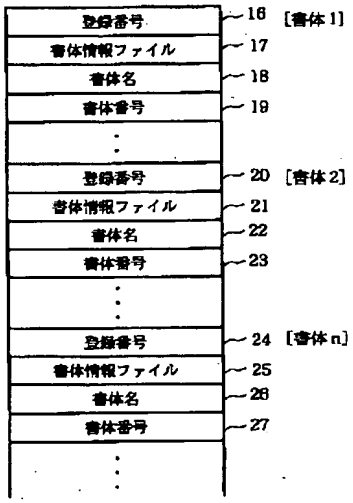


【図４１】

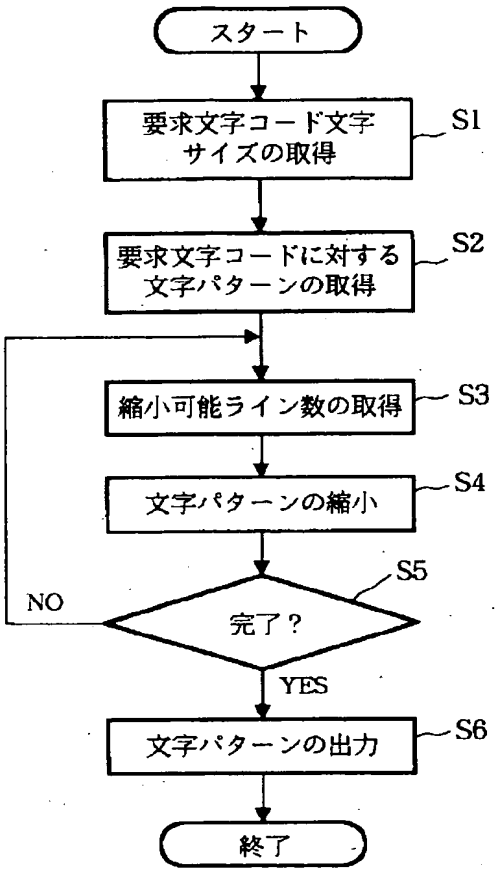
【図２７】


スタート
インストールメディアの設定
S301
インストールメディアより
インストールデータの取得
S302
インストール先の書体情報
を格納して書体情報ファイ
ルの情報読み出し
S303
S304
情報ファイルが
有るか？
NO
S305
書体情報ファイルの作成
YES
書体情報ファイルとインス
トール書体の書体情報を比較
S306
S307
同じ書体データ
が存在するか？
NO
S308
新規の書体インストール処理
YES
更新の書体インストール処理
S309
終了
S310

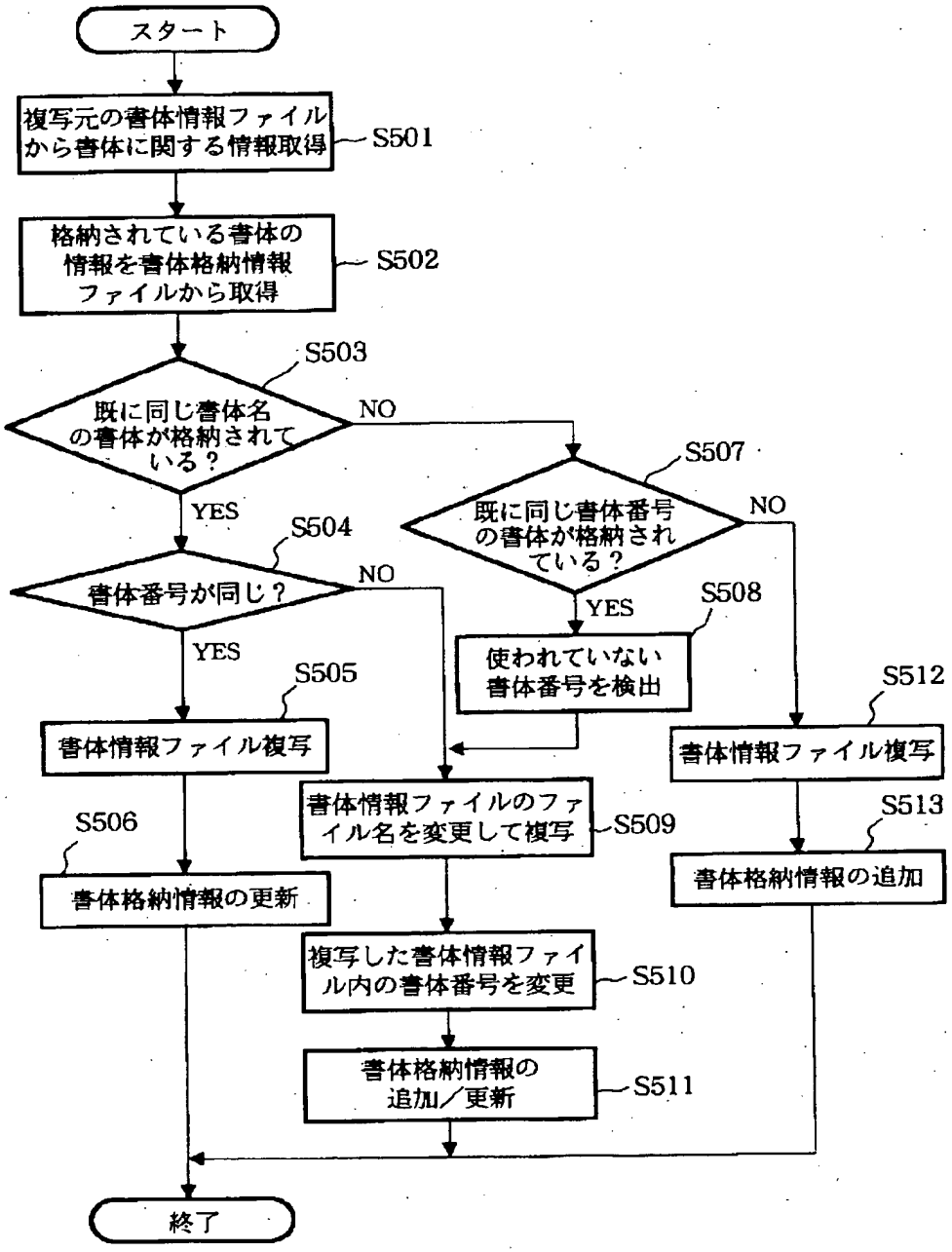
【図４４】


登録番号
16
[書体1]
書体情報ファイル
17
書体名
18
書体番号
19
登録番号
20
[書体2]
書体情報ファイル
21
書体名
22
書体番号
23
登録番号
24
[書体n]
書体情報ファイル
25
書体名
26
書体番号
27

【図３９】


スタート
要求文字コード文字サイズの取得
S1
要求文字コードに対する文字パターンの取得
S2
縮小可能ライン数の取得
S3
文字パターンの縮小
S4
完了？
S5
NO
YES
文字パターンの出力
S6
終了

【図４５】



フロントページの続き

(72)発明者　柏木　正樹
東京都大田区下丸子３丁目30番２号　キヤノン株式会社内

(72)発明者　石黒　健二
東京都大田区下丸子３丁目30番２号　キヤノン株式会社内

(72)発明者　長田　守
東京都大田区下丸子３丁目30番２号　キヤノン株式会社内

(72)発明者　松木　浩
東京都大田区下丸子３丁目30番２号　キヤノン株式会社内